

**DESARROLLO DE UN SISTEMA DE INFORMACIÓN WEB PARA LA
ADOPCIÓN COMPATIBLE Y EL APADRINAMIENTO DE ANIMALES EN
CONDICIÓN DE CALLE PARA LA CIUDAD DE CÚCUTA, APLICANDO
ANÁLISIS INTELIGENTE.**

Autor:

Enyerson Darwin Camero Correa



Directora:

Ph. D. Diana Ruth Martínez Suárez

Universidad de Pamplona
Facultad de Ingenierías y Arquitectura
Departamento de Ingenierías Electrónica, Eléctrica, Sistemas y
Telecomunicaciones, EEST.
Ingeniería de Sistemas
Villa del Rosario, Colombia.
2022

Dedicatoria

“Quiero dedicar el siguiente trabajo de grado:

Por encima de todo a Dios ya que gracias a su voluntad todo es posible.

A mi madre Solange Correa gracias por tu ejemplo y demostrarme que nunca es tarde para salir adelante.

Mi esposa Alix Vanessa Sayago que hay sido mi acompañara de vida y madre de nuestra hija.

A mi hija Zoe Julietta que me motivó, a ser una mejor persona y buscar un mejor futuro.

Finalmente quiero dedicar esta tesis a todas las personas que directa o indirectamente me apoyaron y extendieron su mano amiga en momentos difíciles”

Agradecimientos

Agradezco a todas las personas que me apoyaron, acompañaron, dieron fuerzas y me inspiraron en este proceso de formación.

De manera especial a mi tutora Diana Ruth Martínez por su dedicación, paciencia y comprensión.

A todos los profesores que hicieron parte de mi rol estudiantil por compartirme sus conocimientos.

A toda mi familia en especial a mi madre Solange Correa la cual me acompaña en todo momento y es el mayor apoyo que tengo en mi vida.

A mi compañera de vida Alix Vanessa Sayago que cree en mí y me apoya en cada paso que doy motivándome y dándome razones para seguir adelante.

A mi hija Zoe Julietta que es el motivo para mejorar y crecer profesionalmente.

A mis compañeros y amigos de vida universitaria, en especial a mi amigo Miguel Diaz que desde el primer semestre me brindo su amistad y conocimiento, con el cual vimos muchos amaneceres estudiando y adquiriendo conocimiento

Resumen

Promover la investigación en innovación en el área de bienestar animal, es importante porque se ha demostrado que los animales generan un impacto positivo en la vida de las personas y son muy importantes en el desarrollo social, intelectual y afectivo de los niños lo cual se ve reflejado positivamente en sus etapas posteriores, especialmente en su relación armoniosa con otros seres humanos. Los adultos mayores que por diversas razones se van quedando solos, encuentran en las mascotas una gran compañía con la que expresan sus sentimientos y emociones y alejan los fantasmas de la soledad y la depresión. (Álvarez, 2016). En la presente investigación, se desarrolló un sistema de información web para la adopción compatible y el apadrinamiento de animales en condición de calle para la ciudad de Cúcuta, utilizando las herramientas de Golang y React.js. La idea es que el sistema sea una fuente de datos que permita a las fundaciones tener cifras más exactas sobre los animales que son rescatados y cuales son adoptados, las cuales a su vez contribuirán con la planeación y acciones para ayudar a los animales en condición de calle. Es importante señalar que, entre los resultados de este proyecto, está la mejora de la visibilidad del trabajo realizado por las fundaciones, el registro de los animales adoptados y la facilitación del proceso de donación. Adicionalmente, cabe resaltar también el impacto social ya que con este sistema se promueve la tenencia responsable de mascotas, la cual se podrá ver reflejada en la disminución de animales en condición de calle, evitando que puedan morder personas, transmitir enfermedades o generar accidentes de tránsito, y se podrán canalizar ayudas a las diferentes fundaciones que generan campañas en pro del bienestar animal.

Palabras claves: Go, framework Iris, proceso de adopción, mascota.

Abstract

Promoting innovation research in the area of animal welfare is important because it has been shown that animals generate a positive impact on people's lives and are very important in the social, intellectual and affective development of children, which is seen positively reflected in his later stages especially in his harmonious relationship with other human beings. Older adults who for various reasons are left alone, find in pets a great company with which they express their feelings and emotions and drive away the ghosts of loneliness and depression. (Alvarez, 2016). In the present investigation, a web information system was developed for the compatible adoption and sponsorship of street animals for the city of Cúcuta, using the Golang and React.js tools. The idea is that the system is a source of data that allows foundations to have more exact figures on the animals that are rescued, and which are adopted, which in turn will contribute to planning and actions to help homeless animals. It is important to point out that, among the results of this project, is the improvement of the visibility of the work carried out by the foundations, the registry of the adopted animals and the facilitation of the donation process. Additionally, it is also worth noting the social impact since this system promotes responsible pet ownership, which may be reflected in the reduction of animals living on the street, preventing them from biting people, transmitting diseases or generating traffic accidents, and aid can be channeled to the different foundations that generate campaigns in favor of animal welfare.

Keywords: Go, Iris framework, adoption process, pet.

Tabla de Contenido

Problema de la Investigación.....	10
Planteamiento del Problema	10
Justificación	11
Objetivos.....	12
Objetivo general	12
Objetivos específicos.....	12
Acotaciones	12
Marco Teórico.....	13
Antecedentes.....	13
Antecedentes Nacionales.....	13
Antecedentes Internacionales 	13
Bases Teóricas	15
¿Qué es una mascota?	15
¿Qué es la adopción?	15
Inteligencia Artificial.....	16
Algoritmo K-Nearest Neighbor(KNN).....	16
Metodología ágil de desarrollo Kanban	17
Arquitectura API REST.....	23
Marco Legal.....	25
Ley estatutaria 1581 de 2012.....	25
Resolución 512 de 2019	25
Desarrollo del Proyecto De Investigación	26
Herramientas de Desarrollo	26
Go o Golang	26

Framework de Iris.....	27
React.js	27
MongoDB	27
GoLearn.....	28
Requerimientos del sistema	29
Requerimientos funcionales	29
Diagrama de caso de uso	50
Diseño del Sistema	53
Diagrama de dominio	53
Diseño de la base de datos.....	54
Diagrama Entidad Relación.....	55
Modelo Documental	56
Modulo inteligente.....	86
Conclusiones.....	109
Recomendaciones.....	110
Bibliografía.....	110
Anexo a. Manual del usuario.....	113

Lista de Ilustraciones

<i>Ilustración 1 - Ejemplo Algoritmo KNN</i>	17
<i>Ilustración 2 - Tablero Kanban</i>	17
<i>Ilustración 2 tablero - Kanban GitHub</i>	19
<i>Ilustración 3 - Tarjeta Kanban</i>	20
<i>Ilustración 4 - Historia de usuario</i>	21
<i>Ilustración 5-Arquitectura REST</i>	23
<i>Ilustración 6 Caso de uso creación de usuarios</i>	33
<i>Ilustración 7 Caso de uso autenticar usuarios</i>	34
<i>Ilustración 8 caso de uso modificar información del perfil de usuario</i>	35
<i>Ilustración 9 Caso de uso modificar contraseña del usuario</i>	36
<i>Ilustración 10 Caso de uso: Eliminar usuario</i>	37
<i>Ilustración 11 Caso de uso CRUD de animales</i>	38
<i>Ilustración 12 Caso de uso: Listar animales</i>	39
<i>Ilustración 13 Caso de uso CRUD de Razas</i>	40
<i>Ilustración 14 Caso de uso Consultar animales para adopción.</i>	41
<i>Ilustración 15 Caso de uso Buscar animales</i>	41
<i>Ilustración 16 Caso de uso: Enviar solicitud de adopción.</i>	42
<i>Ilustración 17 Caso de uso: Modificar la solicitud de adopción</i>	43
<i>Ilustración 18 Caso de uso: Listar solicitudes de adopción</i>	44
<i>Ilustración 19 Caso de uso: Consultar datos de solicitud de adopción.</i>	44
<i>Ilustración 20 Caso de uso: Listar adopciones</i>	45
<i>Ilustración 21 Caso de uso: Aprobar adopción</i>	46
<i>Ilustración 22 Caso de uso: Sugerir animales según el perfil del interesado en la adopción.</i>	47
<i>Ilustración 23- Caso de uso Ver datos básicos del animal</i>	47
<i>Ilustración 24 Caso de uso: Realizar donación en línea</i>	48
<i>Ilustración 25 Caso de uso: Registrar apadrinamiento.</i>	49

<i>Ilustración 25 Diagrama caso de uso general</i>	50
<i>Ilustración 26 Diagrama Dominio</i>	53
<i>Ilustración 27 Diagrama Entidad Relación</i>	55
<i>Ilustración 28 Diagrama Colecciones en MongoDB</i>	56
<i>Ilustración 29 Documento usuario</i>	57
<i>Ilustración 30 Documento animal</i>	58
<i>Ilustración 31-Documento Especies</i>	59
<i>Ilustración 32-Documento solicitudes de adopciones</i>	60
<i>Ilustración 33 Documento adoptante</i>	61
<i>Ilustración 34 Documento fundación</i>	62
<i>Ilustración 35-Crear proyecto mongoDB</i>	63
<i>Ilustración 36-Proyecto mongoDB</i>	63
<i>Ilustración 37-Servicios mongoDB</i>	64
<i>Ilustración 38-Proveedor y región mongoDB</i>	64
<i>Ilustración 39-Creación de usuario mongoDB</i>	65
<i>Ilustración 40-Clúster mongoDB</i>	65
<i>Ilustración 41-Creación Base de Datos MongoDB</i>	66
<i>Ilustración 42-Base de Datos mongoDB</i>	66
<i>Ilustración 43-Diseño Arquitectura de Apis REST</i>	67
<i>Ilustración 44-archivo variables de entorno back-end</i>	67
<i>Ilustración 45-archivo variables de entorno front-end</i>	68
<i>Ilustración 47-Estructura de la aplicación</i>	69
<i>Ilustración 48-Asignación de endpoints</i>	70
<i>Ilustración 49-Archivo main.go</i>	71
<i>Ilustración 50-Dependencias en Golang</i>	71
<i>Ilustración 51-Directorio front-end</i>	72
<i>Ilustración 52-Dependencias front-end</i>	73
<i>Ilustración 53-Página de inicio</i>	74
<i>Ilustración 54-Animales en adopción</i>	74

<i>Ilustración 55-Paginación de animales disponibles.....</i>	<i>75</i>
<i>Ilustración 56-Medios de donación</i>	<i>76</i>
<i>Ilustración 57-Detalle Animal.....</i>	<i>76</i>
<i>Ilustración 58-Solicitud de adopción</i>	<i>77</i>
<i>Ilustración 59-Formulario de apadrinamiento</i>	<i>77</i>
<i>Ilustración 60- Animales de la fundación</i>	<i>78</i>
<i>Ilustración 61- Detalle animal</i>	<i>78</i>
<i>Ilustración 62-Vacunas del Animal.....</i>	<i>79</i>
<i>Ilustración 63 - Control de peso del animal.....</i>	<i>79</i>
<i>Ilustración 64- Agregar animal.....</i>	<i>80</i>
<i>Ilustración 65- Eliminar animal</i>	<i>80</i>
<i>Ilustración 66- Solicitudes de adopción</i>	<i>81</i>
<i>Ilustración 67- Detalle de la solicitud de adopción</i>	<i>81</i>
<i>Ilustración 68- Cambiar estado de la solicitud de adopción.....</i>	<i>82</i>
<i>Ilustración 69-Opciones para donar</i>	<i>83</i>
<i>Ilustración 70- Formulario de donación</i>	<i>83</i>
<i>Ilustración 71- Opciones de pago mercado pago.....</i>	<i>84</i>
<i>Ilustración 72 - Pago con tarjeta de crédito</i>	<i>84</i>
<i>Ilustración 73- Dato del titular de la tarjeta de crédito.....</i>	<i>85</i>
<i>Ilustración 74- Selección de numero de cuotas</i>	<i>85</i>
<i>Ilustración 75- Confirmar pago</i>	<i>86</i>
<i>Ilustración 76- Transacción realizada con éxito en mercado pago</i>	<i>86</i>
<i>Ilustración 78 - importar librerías golearn</i>	<i>87</i>
<i>Ilustración 79 - información de aprendizaje.....</i>	<i>88</i>
<i>Ilustración 80- Función de búsqueda parte 1</i>	<i>88</i>
<i>Ilustración 81- Función de búsqueda parte 2</i>	<i>89</i>
<i>Ilustración 82 - servicio búsqueda sugerida</i>	<i>89</i>
<i>Ilustración 83 - Controlador búsqueda sugerida</i>	<i>90</i>
<i>Ilustración 84 - Ingresar sugerir mascota.....</i>	<i>91</i>

<i>Ilustración 85 - Formulario para sugerencia de mascota</i>	<i>91</i>
<i>Ilustración 86 - animal sugerido.....</i>	<i>92</i>
<i>Ilustración 87- Controlador en Iris.....</i>	<i>93</i>
<i>Ilustración 88-Modelo en Golang.....</i>	<i>94</i>
<i>Ilustración 89 - Servicio en Golang</i>	<i>95</i>
<i>Ilustración 90- Características del servicio</i>	<i>95</i>
<i>Ilustración 91- Consulta MongoDB desde Golang.....</i>	<i>96</i>
<i>Ilustración 92 - Inyección de dependencias.....</i>	<i>97</i>
<i>Ilustración 93 - Definición de rutas.....</i>	<i>98</i>
<i>Ilustración 94 - Obtener token en login.....</i>	<i>99</i>
<i>Ilustración 95 - Información del token</i>	<i>99</i>
<i>Ilustración 96 - Generación del token.....</i>	<i>100</i>
<i>Ilustración 97 - Pruebas de velocidad de Iris</i>	<i>101</i>
<i>Ilustración 98 - Velocidad de una solicitud de la aplicación</i>	<i>101</i>
<i>Ilustración 99 – Primer resultado de la encuesta de aceptación de la aplicación</i>	<i>103</i>
<i>Ilustración 100- Segundo resultado de la encuesta de aceptación de la aplicación</i>	<i>104</i>
<i>Ilustración 101 - Tercer resultado de la encuesta de aceptación de la aplicación</i>	<i>104</i>
<i>Ilustración 102 – Cuarto resultado de la encuesta de aceptación de la aplicación.....</i>	<i>105</i>
<i>Ilustración 103 – Quinto resultado de la encuesta de aceptación de la aplicación.....</i>	<i>105</i>
<i>Ilustración 104 - Sexto resultado de la encuesta de aceptación de la aplicación.....</i>	<i>106</i>
<i>Ilustración 105 - Séptimo resultado de la encuesta de aceptación de la aplicación</i>	<i>106</i>
<i>Ilustración 106 - Octavo resultado de la encuesta de aceptación de la aplicación</i>	<i>107</i>
<i>Ilustración 107 - Noveno resultado de la encuesta de aceptación de la aplicación</i>	<i>107</i>

Lista de Tablas

<i>Tabla 1 – Requerimientos funcionales</i>	<i>30</i>
<i>Tabla 2 - WSJF</i>	<i>32</i>
<i>Tabla 3 Registrar usuario</i>	<i>33</i>
<i>Tabla 4 RF-2 Autenticar usuario.....</i>	<i>34</i>
<i>Tabla 5 RF-3 Modificar información del perfil de usuario.</i>	<i>35</i>
<i>Tabla 6 RF-4 Modificar contraseña del usuario.....</i>	<i>36</i>
<i>Tabla 7 RF-5 Eliminar usuario.</i>	<i>37</i>
<i>Tabla 8 RF-6 CRUD animales</i>	<i>38</i>
<i>Tabla 9 RF-7 Listar animales.</i>	<i>39</i>
<i>Tabla 10 RF-8 CRUD Razas</i>	<i>39</i>
<i>Tabla 11 RF-10 Consultar animales para adopción.....</i>	<i>40</i>
<i>Tabla 12 RF-11 Buscar animales</i>	<i>41</i>
<i>Tabla 13 RF-12 Enviar solicitud de adopción.....</i>	<i>42</i>
<i>Tabla 14 RF-14 Modificar la solicitud de adopción</i>	<i>43</i>
<i>Tabla 15 RF-15 Listar solicitudes de adopción.....</i>	<i>43</i>
<i>Tabla 16 RF-16 Ver información de la solicitud de adopción</i>	<i>44</i>
<i>Tabla 17 RF-18 Listar adopciones.....</i>	<i>45</i>
<i>Tabla 18 RF-19 Aprobar Adopción</i>	<i>46</i>
<i>Tabla 19 RF-20 Sugerir animales.....</i>	<i>46</i>
<i>Tabla 20- RF- 22 - Ver datos básicos del animal.....</i>	<i>47</i>
<i>Tabla 21 RF-21 Realizar donación en línea.</i>	<i>48</i>
<i>Tabla 22 RF-21 Realizar donación en línea.</i>	<i>49</i>
<i>Tabla 22 RNF-1 Roles</i>	<i>51</i>
<i>Tabla 23 RNF-2 Tiempo de Repuesta</i>	<i>51</i>
<i>Tabla 24 RNF-3 Amigable con el usuario.....</i>	<i>51</i>
<i>Tabla 25-Diccionario de Datos Colección Usuarios</i>	<i>56</i>

<i>Tabla 26- Diccionario de Datos Colección Animal</i>	<i>58</i>
<i>Tabla 27- Diccionario de Datos Colección Especies</i>	<i>59</i>
<i>Tabla 28- Diccionario de Datos Colección solicitudes de adopción</i>	<i>59</i>
<i>Tabla 29- Diccionario de Datos Documento Adoptante</i>	<i>60</i>
<i>Tabla 30-Diccionario de datos Colección apadrinamientos</i>	<i>61</i>

Problema de la Investigación

Planteamiento del Problema

No se conocen cifras oficiales, pero se estima que en Colombia existe cerca de un millón de animales abandonados, solamente en la ciudad de Bogotá, el instituto Distrital de Protección y Bienestar Animal asegura que más de 33 mil animales fueron abandonados desde el 2010 y 2016. Cúcuta no es ajena a esta problemática ya que se cree que hay entre mil o dos mil animales en las calles. Un gran número de animales en situación de desamparo con una calidad de vida pésima, sin techo ni comida, tal como se habla en un artículo del diario la Opinión, que también menciona: que es un problema evitable con más información y concientización. La participación de la sociedad es muy importante para cambiar las cifras de abandono y ayudar a solucionar el problema. (Oliver, 2020).

Existen algunas fundaciones que se encargan de ayudar a los animales de calle, pero adicional a sus problemas económicos, tienen un proceso de adopción ineficiente pues lo realizan por medio de la publicación de la información de las mascotas en redes sociales. Estas publicaciones no son muy visibles pues se ven opacadas por la gran cantidad de información que maneja este medio.

Otro problema, tiene que ver con el hecho de que no se lleva un control y seguimiento de los animales adoptados, lo que hace imposible tener un registro de la función de la fundación para la gestión de ayudas o saber si sus adoptantes cumplen con las características para brindar un buen hogar a sus mascotas.

De toda esta problemática surge la pregunta de investigación:

¿Será posible mejorar el proceso de adopción de animales en situación de abandono utilizando las nuevas tecnologías?

Justificación

El sistema Web en las fundaciones es una iniciativa para promover la adopción y apadrinamiento de animales, con el fin de que las personas puedan ponerse en contacto más fácilmente con las fundaciones ya sea con la intención de adoptar o apadrinar animales o hacer donativos, ya que estas tienen a su cargo varios animales que generan costos de funcionamiento y que solo se sólo se pueden cubrir a través de esta forma. De acuerdo a lo anterior el desarrollo de una aplicación Web es de vital importancia para que las fundaciones se enfoquen en el cuidado de las mascotas y es indispensable para su funcionamiento permitiéndoles volverse más eficientes y contribuyendo así con el mejoramiento de la calidad de vida de los animales de calle.

Objetivos

Objetivo general

Desarrollar un sistema de información web para la adopción compatible y el apadrinamiento de animales en condición de calle para la ciudad de Cúcuta, aplicando análisis inteligente.

Objetivos específicos

- Analizar el proceso de adopción de mascotas y apadrinamiento de mascotas.
- Definir los requerimientos para el desarrollo del sistema.
- Diseñar la base de datos y arquitectura del sistema.
- Construir los módulos de la aplicación usando el Framework Go Iris, y la librería React js.
- Validar el software con una Fundación que se encargue del cuidado de animales en condición de calle en la ciudad de Cúcuta.

Acotaciones

- La aplicación web se realizará con el lenguaje Go V.1.3 y Framework Iris v12.2.0
- La base de datos se implementará en Mongo 6.0.1
- Se validará con una fundación de cuidado de animales en condición de calle de la ciudad de san José de Cúcuta
- La aplicación estará alojada en la en un servidor vps.

Marco Teórico

Antecedentes

Los siguientes antecedentes reflejan el estado actual de las investigaciones realizadas en sistemas que contribuyen en el mejoramiento del proceso de adopción de mascotas, sirviendo como punto de partida para el diseño y construcción del sistema propuesto en este proyecto.

Antecedentes Nacionales

(Méndez Melo, Villafañe Castaño, & Martínez España, 2019) titulado **“aplicación móvil para la adopción de mascotas abandonadas”**, Este trabajo tiene como objetivo la creación de una aplicación móvil para dispositivos Android, que permita a las personas acceder a la información sobre la hoja de vida de los animales desprotegidos y poder realizar la adopción en línea.

Antecedentes Internacionales

En un primer trabajo de (Cabrera Paucar, 2017) titulado **“desarrollo de una aplicación web de adopción de mascotas en la ciudad de Loja Ecuador”**, el trabajo describe el desarrollo de una aplicación web usando el framework Django para la adopción de mascotas en Loja (PetHome), el cual permite mejorar los procesos en favor de las mascotas como: adopción, búsqueda de mascotas extraviadas y eventos. El proceso de desarrollo se realizó mediante una serie de pasos como la recolección de información, utilizando técnicas como la entrevista y la encuesta. Este trabajo demostró la factibilidad del proyecto y un gran número de herramientas disponibles para su planificación y realización, que detrás de los animales hay una gran cantidad de personas dispuestas a ayudar de diferentes maneras y proporcionando las herramientas correctas, serían mayores las ayudas para los animales

En un segundo trabajo de (Toapanta Gualpa, 2019) llamado “**desarrollo de un sistema de gestión sectorizado de adopción de mascotas para la fundación camino a casa**”. El cual fue elaborado para plataforma web con el fin de optimizar el proceso de adopciones y rescates de mascotas para la ciudad de Quito. Fundación Camino a Casa tenía Facebook, pero se encontraron con el problema de que cuando se trata de adoptar mascotas por este medio impera el desorden y adopción termina tardando demasiado tiempo, ya que no se lleva la cuenta de adopciones por categorías, como la raza, el lugar de procedencia, la edad, no se puede identificar los anuncios con adopciones exitosas, no se lleva un historial de conversaciones con las personas interesadas, no se puede hacer un filtrado ni búsqueda por nombre y tampoco en una forma para mostrar las adopciones exitosas. La importancia de estos datos ayudará a la fundación a generar estrategias que contribuyan al cuidado y mejoramiento de las vidas de las mascotas a su cuidado.

En un tercer trabajo consultado. Realizado por (Zorita Bote, 2016) titulado: **Aplicación gestión de mascotas OURPETS**. quien dice: La construcción de una aplicación móvil, nativa en Android, que pretende facilitar la gestión del historial veterinario para usuarios con mascotas, facilitando su acceso a la información sobre la salud de su mascota siempre que se requiera. También las instituciones privadas y públicas que alberguen las mascotas dispondrán de las opciones para enriquecer las hojas de vida y presentar al público de manera controlada toda la información de interés para la toma de decisiones y consumo de servicios disponibles, seguimiento y control de estos.

En un quinto trabajo consultado. Realizado por (Zaldumbide Burbano, 2018) titulado: **Desarrollo de una aplicación web de red social para estimular los vínculos**

afectivos entre mascotas y sus dueños que decía: El amor animal ha brindado beneficios a los seres humanos a lo largo de la historia. La estimulación de estos vínculos afectivos entre personas y mascotas brinda ventajas no solo físicas, emocionales y espirituales, sino también evolutivas. Existen en el mercado actualmente varias aplicaciones para los animales de compañía, pero no con el enfoque y el objetivo de la solución aquí presente. Pet Lovers, es una aplicación web de red social para estimular los vínculos afectivos entre mascotas y sus dueños. En el presente documento se muestra su desarrollo, es decir, su marco teórico, el desarrollo de la solución y la validación de esta. Gracias a la aplicación las personas podrán interactuar con otras y con las mascotas de estas, creando ambientes propicios para utilización de los beneficios que estos vínculos representan. La solución fue realizada con el framework Scrum, comprende una aplicación web programada en PHP, con la ayuda del framework Laravel con base de datos MySQL y combinado con tecnologías para la mensajería instantánea como Pusher y Vue.js. Utilizando el patrón Modelo, Vista y Controlador.

Bases Teóricas

¿Qué es una mascota?

Son animales domésticos cuya principal función es la de acompañar al ser humano, incluso llegando a ser tratados como un miembro más de la familia. La mascota ayuda a disminuir el sentimiento de soledad y trae un ambiente más agradable al hogar. En algunos casos las mascotas pueden llegar a ejecutar tareas, con el entrenamiento adecuado, con la finalidad de facilitar la vida de las personas.

¿Qué es la adopción?

La adopción permite brindarle un nuevo hogar a mascotas que se encuentran en una fundación, en el cual ha pasado por un proceso de recuperación tras ser abandonado

por sus anteriores dueños. La adopción debe ser realizada por personas que puedan suplir las necesidades de la mascota, como lo son techo y comida, entre otros recursos necesarios para el bienestar de las mascotas.

Inteligencia Artificial

AI se compone de dos palabras, Artificial e Intelligence, donde Artificial define "*hecho por el hombre*" e Intelligence define "capacidad de pensamiento", por lo que AI significa "capacidad de pensamiento artificial".

La inteligencia artificial es una disciplina de las Ciencias de la Computación cuyo objetivo es el desarrollo de sistemas que pueden actuar de una manera inteligente. Este tipo de sistemas son capaces de aprender a partir de una serie de datos. (Fundación Aquae, 2022)

Algoritmo K-Nearest Neighbor(KNN)

Nearest Neighbor es uno de los algoritmos de aprendizaje automático más simples basado en una técnica de aprendizaje supervisado. La similitud entre el nuevo dato y los casos disponibles. Coloca el nuevo caso en la categoría más cercana a las categorías disponibles.

El algoritmo K-NN almacena todos los datos disponibles y clasifica un nuevo punto de datos en función de la similitud. Esto significa que cuando aparecen nuevos datos, se pueden colocar fácilmente en una categoría bien definida utilizando el algoritmo K-NN.

En la fase de entrenamiento, el algoritmo KNN simplemente almacena el conjunto de datos y cuando recibe nuevos datos, los clasifica en una categoría que es muy similar a los nuevos datos. (Java Point, 2022).



Ilustración 1 - Ejemplo Algoritmo KNN

Fuente (Java Point, 2022)

Metodología ágil de desarrollo Kanban

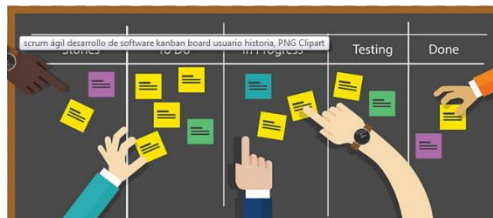


Ilustración 2 - Tablero Kanban

Kanban ofrece un esquema visual donde se puede ver fácilmente el estado del trabajo por medio de un tablero donde cada tarjeta representa una tarea. Esta disposición permite establecer unas metas asequibles equilibrando el flujo de trabajo. Al cerrar cada una de estas metas genera una sensación de satisfacción en el usuario lo que lo motiva y permite un mejor desarrollo de las actividades como se aprecia en la Ilustración 2 - Tablero Kanban

Medición del rendimiento. Con el método Kanban se puede medir el rendimiento de los trabajadores o equipos, así como detectar cualquier problema que se genere durante el transcurso de la tarea.

Organización del flujo de trabajo. El flujo de trabajo queda claro en cada fase del proyecto gracias a la organización de las tareas por estados y de la duración estimada que se indica para cada una de ellas. Por ello, es posible prever una cantidad de trabajo adecuada y asequible para el cumplimiento de las metas.

Cumplir los tiempos de entrega. Ligado al segundo punto, gracias a la organización del flujo de trabajo podremos medir y cumplir con los tiempos de entrega demandados por el cliente.

Distribución de tareas. Poder ver las tareas que se han hecho, las que se están haciendo y las que quedan por hacer. Ayuda a una correcta distribución: todo el equipo puede ver cómo va el transcurso del trabajo y saber cuál es el siguiente paso.

Mejorar la calidad del producto. Con buena organización y con un buen control de los tiempos, se podrán detectar posibles problemas y, por consecuencia, encontrar la solución.

Evitar la acumulación de trabajo. Al prever todas las tareas previstas y el tiempo estimado para su cumplimiento, el equipo podrá organizarse y asignar cada trabajo sin sobrecargar. (Kanban Tool., s.f.)

Los elementos que conforman Kanban son los siguientes:

Tablero Kanban: El tablero Kanban es una pizarra que bien puede ser física o digital, en la cual se visualiza el estado de las tareas, todas las tareas previamente ya definidas. El tablero tiene cuatro columnas (por hacer, desarrollando, probando, realizada) que representan el estado de cada tarea a realizar. Todas las tareas al iniciar se sitúan en la columna “Por hacer”, cuando una tarea es tomada para su realización debe

ser pasada a la columna de “Desarrollando” y finaliza cuando dicha tarea se ha realizado tanto en la lógica detrás de su implementación en el código (back-end) tanto la interfaz de usuario (front-end) para luego ser pasada a la columna donde se realizarán las diferentes pruebas que garantizarán el correcto funcionamiento de la tarea realizada. Si la tarea aprueba las diferentes pruebas realizadas su estado debe ser pasado a la columna realizada. Si por lo contrario la tarea presenta algún error en las pruebas debe ser devuelta a la columna “desarrollando” para realizar los respectivos ajustes y volver a ser probada para su aprobación, esté flujo puede repetirse en varias ocasiones hasta que la tarea cumpla los criterios para su aprobación.

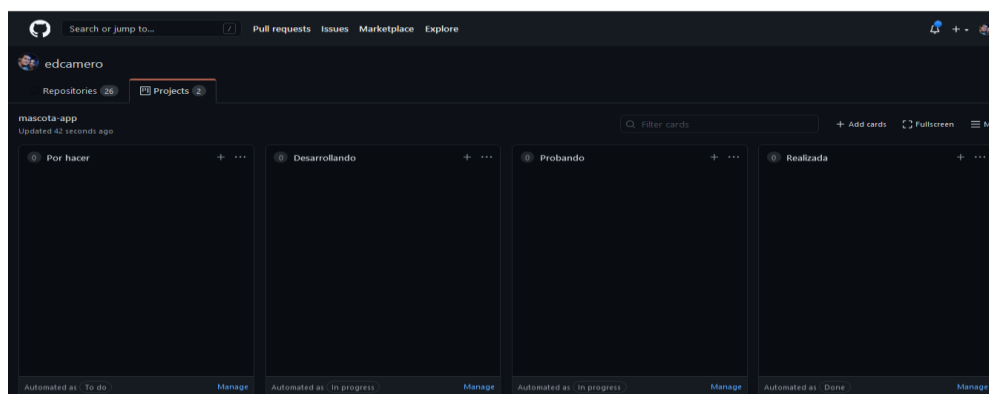


Ilustración 3 tablero - Kanban GitHub

Se escoge como tablero Kanban uno proporcionado por GitHub (ver Ilustración 3 tablero - Kanban GitHub) para los desarrolladores que suben sus proyectos en los repositorios de GitHub, en el cual solo se agrega la columna probando, ya que las otras vienen por defecto en el tablero al crear un nuevo proyecto en GitHub.

Al tratarse de un proyecto desarrollado por una sola persona; en las columnas “Desarrollando” y “Probando” no debe haber más de una tarea, ya que esto permite a la

persona que la está realizando enfocarse en una sola tarea y una mayor eficiencia en su realización.

Al llevar las tareas a la columna “Realizada” la persona puede sentir una liberación de endorfina, que le está dando el cierre de esta actividad los que es bueno y motiva en la continuación de las demás tareas. Haciendo más sencilla y motivante la realización de más tareas.

Tarjeta Kanban: Las tarjetas Kanban representan las tareas planeadas (ver Ilustración 4 - Tarjeta Kanban), y estas pueden moverse a la siguiente columna, exceptuando en la columna “Realizada” donde ya la tarjeta no debe moverse, y en la columna “probando” donde la tarjeta puede regresar a la columna anterior.

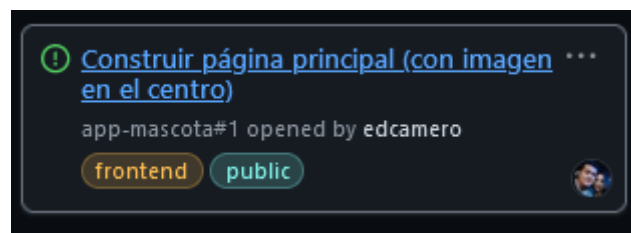


Ilustración 4 - Tarjeta Kanban

En la tarjeta Kanban se agrega el nombre de la tarea y la persona asignada a realizarla, y como se puede ver en la imagen se pueden agregar etiquetas en la tarea para dar un contexto a su realización.

Historia de usuario Kanban: Una historia de usuario está relacionada con la tarjeta Kanban, pero en la historia de usuario se puede encontrar información más detallada y que aporta un valor a los usuarios. En la información que podemos encontrar en la historia de usuario están campos como: id, nombre, descripción, valor, estimación, criterio de validación y prioridad como se puede apreciar en la Ilustración 5 - Historia de usuario.

ID:	RF-1	Nombre:	Registrar usuario
Valor:	21	Prioridad:	1
Descripción	Como usuario deseo registrarme en el sistema con mi respectivo rol (adoptante, fundación, admin) para hacer uso de las funcionalidades disponibles.		
Criterio de validación:	Cuando los datos son correctos el sistema permitirá la creación del nuevo usuario. Los adoptantes pueden registrase ellos mismos en el sistema. Las fundaciones haran un pre-registro de los datos solicitados y el administrador validara dicha información y aprobara a la fundación segun corresponda. los administradores tendran un formulario que permite agregar a otro administrador.		

Ilustración 5 - Historia de usuario

Id: Identificación de la historia de usuario.

Nombre: nombre de la historia de usuario y es el que se agrega en la tarjeta Kanban para visualizar su estado en el tablero Kanban.

Descripción: Comunica a la persona que desarrollará dicha historia de usuario información detallada, tratando de responder siempre a las siguientes preguntas: ¿Quién se beneficia? ¿Qué se quiere? y ¿Cuál es el beneficio? Mike Cohn recomienda seguir el siguiente patrón que garantiza que la funcionalidad está descrita a un alto nivel (Cohn, 2009) y de una manera no demasiado extensa:

Cómo [rol del usuario], quiero [objetivo], para poder [beneficio]

Cómo [rol del usuario]: ¿Quién es la persona que usa la función para quién estamos construyendo esto? No se trata de un cargo o un rol profesional, se trata del tipo de persona que hay detrás de la necesidad descrita, personaje que describimos con la User-Persona y que permite identificar de quién es y entender cómo trabaja, cómo piensa y siente, en definitiva, tener empatía por él.

Quiero [objetivo]: Aquí estamos describiendo el objetivo del usuario, no la función que se creara. ¿Qué es lo que realmente quiere lograr?

Para poder [beneficio]: ¿Qué se está solucionado con lo que se está haciendo?
¿Cuáles son las necesidades del usuario? ¿Cuál es el beneficio general que está tratando de lograr?

Valor: valor que aporta la historia de usuario al proyecto una vez es finalizada, está fuertemente relacionada con la lógica del negocio y es de gran importancia al momento de priorizar las historias de usuario, hay diferentes formas de establecer su valor por lo general se hace con valores numéricos, los cuales los establece el líder de proyecto o miembros de equipo se sienta más cómodos, así como la escala de los valores los cuales pueden ser de 1 a 10, de 1 a 100, o la serie de Fibonacci la cual fue escogida por el hecho que la diferencia de cada uno de sus números de la serie refleja con una mejor proporción la incertidumbre entre historias de usuario para historias de valor 1, 2 o 3 son más comprendidas que historias cuyo valores son más elevados como 21, 42 y 63.

Estimación. Es el esfuerzo de en el desarrollo necesario para dar por terminada una historia de usuario, según los miembros de equipo se puede utilizar una unidad de tiempo bien definida como horas, días o una unidad mucho más arbitraria pero que pueda representar de forma clara el esfuerzo para dicha historia de usuario.

Prioridad: Permite determinar el orden en que las historias de usuario son tomadas para su desarrollo. Y para calcular su valor se usa la fórmula **WSJF (Weighted Shortest Job First)**.

$$ROI = \frac{Valor + Critica en el tiempo + Reduccion de riesgo}{tamaño}$$

Valor: como se mencionó antes es un valor estimado por el equipo a la respectiva historia de usuario.

Tamaño: Se usa la complejidad y tamaño estimado de la historia de usuario para calcular un tamaño que se aproxime a la realidad lo más posible.

Criticidad en el tiempo: Esta medida se ocupa de la "necesidad" de la entrega del elemento en una escala de tiempo, viene asociado con la caída de valor con el tiempo. A más "necesidad" de entrega más alto será el valor.

Reducción del riesgo: Este valor representa el impacto en el contexto de mejoramiento que recibe el usuario al completarse una historia de usuario, ya que con la realización de está relacionada con la desaparición de una o más riesgos para el dueño del producto.

Arquitectura API REST

La arquitectura API REST proporciona a los desarrolladores un conjunto de restricciones arquitectónicas, que al aplicarla como un todo facilita la escalabilidad de las interacciones de los componentes, usar la generalidad de las interfaces, el uso de componentes independiente de otros componentes. Destaca por aumenta la seguridad y separar responsabilidades. (Reactive Programming, 2022)

Estructura API REST

REST describe 3 conceptos claves los cuales son: Datos, Conectores y Componentes lo cuales será explicados a continuación:

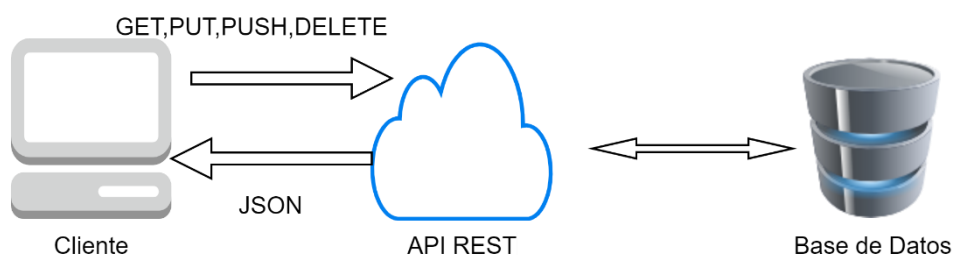


Ilustración 6-Arquitectura REST

Datos: REST propone que los datos sean enviados en bruto (Json) para que sea el interesado de los datos el que los procese para luego ser presentados en la página o cualquier otro servicio que lo solicite. Lo que REST propone no es generar la página web, si no que mandar los datos en bruto para que el consumidor sea el responsable de generar la página a través de los datos que responde REST ver Ilustración 6-Arquitectura REST. (Reactive Programming, 2022)

Conectores: REST establece que todas las solicitudes de conexión son sin estado, es decir, deben contener toda la información necesaria para que la conexión entienda la solicitud, independientemente de las llamadas realizadas antes, por lo que ante la misma solicitud obtenemos la misma respuesta.

Componentes: Por último, los componentes de software específicos que utilizan los complementos para consultar o proporcionar recursos, ya sean aplicaciones cliente como navegadores web (Chrome, Firefox, etc.) o servidores web gratuitos como Apache, IIS, etc. Pero hay algunas partes intermedias que actúan como cliente y servidor a la vez, como, por ejemplo: un proxy o túnel de red, que actúa como servidor al negociar solicitudes, pero como cliente, redirigiendo solicitudes a otro servidor.

Los componentes se pueden confundir con módulos, sin embargo, un módulo es una entidad separada que define cómo se comunican los componentes, mientras que un componente es un software que usa módulos para crear la conversación (Reactive Programming, 2022).

Los servicios REST tienen las siguientes características:

- Hay cuatro operaciones principales que le permiten administrar los recursos: GET para consultar y leer; hacer un POST; PUT para editar; DELETE para eliminar.

- Utiliza hipermedia para permitir que el cliente navegue entre diferentes recursos de la API REST mediante enlaces HTML.
- La respuesta de la API REST suele ser JSON, sin el idioma utilizado.

Marco Legal

Este marco legal brinda información sobre ciertos requisitos y obligaciones legales para asegurar que el desarrollo del proyecto se realice dentro de un marco legal adecuado.

Ley estatutaria 1581 de 2012

En la cual se dictan disposiciones generales para la protección de datos personales la cual abarca el derecho constitucional que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos, y los demás derechos, libertades y garantías constitucionales a que se refiere el artículo 15 de la Constitución Política; así como el derecho a la información consagrado en el artículo 20 de la misma. (funcionpublica.gov.co, 2022)

Resolución 512 de 2019

“Por la cual se adopta la política general de seguridad y privacidad de la información, seguridad digital y continuidad de los servicios del Ministerio/Fondo de tecnologías de la comunicación y las telecomunicaciones y se definen lineamientos frente al uso y manejo de la información” (Mintic, 2022)

Desarrollo del Proyecto De Investigación

En este capítulo se describirán los pasos realizados para la construcción del sistema de información web para la adopción compatible y el apadrinamiento de animales en condición de calle para la ciudad de Cúcuta, aplicando análisis inteligente, realizando los requerimientos y el diseño del sistema partiendo de las necesidades de los usuarios.

Herramientas de Desarrollo

Go o Golang

El lenguaje de programación Go es un proyecto de código abierto para que los programadores sean más productivos.

Go es expresivo, conciso, limpio y eficiente. Sus mecanismos de concurrencia facilitan la escritura de programas que aprovechan al máximo las máquinas multinúcleo y en red, mientras que su novedoso sistema de tipos permite la construcción de programas flexibles y modulares. Go compila rápidamente al código de máquina, pero tiene la conveniencia de la recolección de basura y el poder de la reflexión en tiempo de ejecución. Es un lenguaje rápido, de tipo estático y compilado que se siente como un lenguaje interpretado de tipo dinámico (Go, 2022).

La sintaxis de Golang tiene muchas semejanzas con la sintaxis básica de los lenguajes de la familia C, pero muestra una fuerte influencia de los lenguajes Pascal, Modula y Oberon, desarrollados por Niklaus Wirth. También revela aspectos de lenguajes como Newsqueak y Limbo, que a su vez se inspiran en CSP (Communicating Sequential Processes), de Tony Hoares. Golang es un lenguaje compilado, si bien desde el principio se da gran importancia a una elevada velocidad de traducción. El lenguaje

también cuenta con un recolector de basura automático (Gargabe Collection o GC), que contribuye a **gestionar con eficiencia la memoria** disponible y reduce así los problemas de espacio. (IONOS., 2019)

Framework de Iris

Iris es un marco web rápido, simple, pero con todas las funciones y muy eficiente escrito en Go. Qué proporción una base bellamente expresiva y fácil de usar para su próximo sitio web o API.

La filosofía de Iris es proporcionar herramientas robustas para HTTP, por lo que es una gran solución para aplicaciones de una sola página, sitios web, híbridos o API HTTP públicas. iris es el marco web más rápido jamás creado en términos de rendimiento del mundo real.

Iris no lo obliga a utilizar ningún motor ORM o plantilla específica. Con soporte para los motores de plantillas más potentes y rápidos, puede crear rápidamente la aplicación perfecta (Iris., 2022)

React.js

React ha sido diseñado desde el principio para una adopción gradual, y puede usar tanto React como necesite. Ya sea que desee probar React, agregar algo de interactividad a una página HTML simple o iniciar una aplicación compleja impulsada por React, los enlaces en esta sección lo ayudarán a comenzar. (React JS, 2022)

MongoDB

Es un potente sistema de base de datos no relacional, orientado a documentos y de código abierto. Esto quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección; concepto similar a una tabla de una base de datos relacional, pueden tener esquemas diferentes.

Aunque las colecciones de **MongoDB** no necesitan definir un esquema, es importante que diseñemos nuestra aplicación para seguir uno. Tendremos que pensar si necesitamos normalizar los datos, desnormalizarlos o utilizar una aproximación híbrida. Estas decisiones pueden afectar al rendimiento de nuestra aplicación. En definitiva, el esquema lo definen las consultas que vayamos a realizar con más frecuencia.

MongoDB es especialmente útil en entornos que requieran escalabilidad. Con sus opciones de replicación y sharding, que son muy sencillas de configurar, se puede conseguir un sistema que escale horizontalmente sin demasiados problemas. (MongoDB, 2022).

GoLearn

Ofrece la facilidad de la simplicidad con el poder de la personalización. Cargando los datos como instancias. Luego puede realizar operaciones matriciales en ellos y pasarlos a los estimadores. GoLearn implementa la interfaz de aprendizaje automático, por lo que puede cambiar fácilmente los estimadores por prueba y error. Además, con funciones auxiliares para la división de pruebas y la validación cruzada, GoLearn proporciona los recursos relevantes que los desarrolladores necesitan para administrar la integridad de sus datos. (Golearn, 2022)

Requerimientos del sistema

La definición de los requerimientos son las bases de cualquier proyecto de software, ya que con ellos se identifican las necesidades del problema a solucionar y ayuda a identificar qué elementos se deben agregar en el sistema o por lo contrario que elementos no son necesarios para su funcionamiento. Entre otras ventajas que tiene la creación de requerimientos de manera temprana es la administración de las expectativas, priorizar y asignación de tareas, y estar al tanto del proceso de creación del sistema.

Los requerimientos del presente proyecto fueron obtenidos gracias a los datos obtenidos del cumplimiento del primer objetivo que en un principio tal vez no tenían la estructura de un requerimiento, pero una vez pasados por la fase de análisis permitió ser presentados en una estructura más adecuada.

Requerimientos funcionales

En la siguiente sección se especifican cada uno de los requerimientos funcionales del sistema a tener en cuenta para su funcionamiento, lo cual permite el correcto flujo por las etapas de diseño, planificación, construcción e implementación del sistema.

Los requerimientos están agrupados por módulos, lo que permite mayor organización en el trabajo y entregables más significativos de la aplicación. Los requerimientos funcionales que fueron identificados para el presente proyecto se presentan en la

codigo	nombre
RF-1	Registrar usuario.
RF-2	Autenticar usuario.
RF-3	Modificar información del perfil de usuario.
RF-4	Modificar contraseña del usuario.
RF-5	Eliminar usuario.

RF-6	CRUD animales.
RF-7	Listar animales
RF-8	CRUD razas.
RF-9	Habilitar animal para adoptar
RF-10	Consultar animales para adopción.
RF-11	Buscar animales
RF-12	Enviar solicitud de adopción.
RF-13	Cancelar solicitud de adopción
RF-14	Listar solicitudes de adopción
RF-15	Ver información de la solicitud de adopción
RF-16	Listar adopciones
RF-17	Aprobar Adopción
RF-18	Sugerir animales
RF-19	Ver datos básicos del animal
RF-20	Realizar donación en línea.
RF-21	Sugerir animales según el perfil del interesado en la adopción.
RF-22	Realizar una donación en línea.
RF-23	Apadrinar una mascota.
RF-24	Consultar apadrinamientos.
RF-25	Revocar apadrinamiento.

Tabla 1 – Requerimientos

codigo	nombre
RF-1	Registrar usuario.
RF-2	Autenticar usuario.
RF-3	Modificar información del perfil de usuario.
RF-4	Modificar contraseña del usuario.
RF-5	Eliminar usuario.
RF-6	CRUD animales.
RF-7	Listar animales
RF-8	CRUD razas.
RF-9	Habilitar animal para adoptar
RF-10	Consultar animales para adopción.
RF-11	Buscar animales
RF-12	Enviar solicitud de adopción.

RF-13	Cancelar solicitud de adopción
RF-14	Listar solicitudes de adopción
RF-15	Ver información de la solicitud de adopción
RF-16	Listar adopciones
RF-17	Aprobar Adopción
RF-18	Sugerir animales
RF-19	Ver datos básicos del animal
RF-20	Realizar donación en línea.
RF-21	Sugerir animales según el perfil del interesado en la adopción.
RF-22	Realizar una donación en línea.
RF-23	Apadrinar una mascota.
RF-24	Consultar apadrinamientos.
RF-25	Revocar apadrinamiento.

Tabla 1 – Requerimientos funcionales

La construcción de los requerimientos permite al desarrollador estar enfocado en una tarea a la vez y dándole importancia que le corresponde, cabe recordar que los requerimientos están organizados según el método WSJF como se establece en la metodología de desarrollo Kanban (ver Metodología ágil de desarrollo Kanban) da como resultado Tabla 2 - WSJF.

ID:	código	Nombre	Valor de negocio:	Criticidad en el tiempo	Reducción en el riesgo y valor de oportunidad	Tamaño:	WSJF	Promedio WSJF
modulo usuario	RF-1	Registrar usuario.	21	21	21	21	3,00	2,44
	RF-2	Autenticar usuario.	21	21	13	21	2,62	
	RF-3	Modificar información del perfil de usuario.	2	5	5	5	2,40	
	RF-4	Modificar contraseña del usuario.	2	2	5	5	1,80	
	RF-5	Eliminar usuario.	2	2	8	5	2,40	
modulo animales	RF-6	CRUD animales.	13	13	21	21	2,24	1,75
	RF-7	Listar animales	3	5	5	13	1,00	
	RF-8	CRUD razas.	8	13	21	21	2,00	
modulo adopción	RF-9	Habilitar animal para adoptar	5	8	5	8	2,25	1,97
	RF-10	Consultar animales para adopción.	13	8	8	13	2,23	
	RF-11	Registrar el perfil del interesado en la adopción	8	8	5	13	1,62	
	RF-12	Modificar el perfil del interesado en la adopción	8	2	1	8	1,38	
	RF-13	Solicitar la adopción de un animal.	8	3	8	8	2,38	
	RF-14	Modificar la solicitud de adopción por parte del adoptante	5	5	5	8	1,88	

	RF-15	Listar solicitudes de adopción para la fundación.	5	5	5	13	1,15	
	RF-16	Consultar datos de solicitud de adopción.	5	1	5	5	2,20	
	RF-17	Registrar información de las visitas realizadas por parte de la fundación.	5	5	13	13	1,77	
	RF-18	Generar reporte de solicitudes de adopción.	1	2	2	5	1,00	
	RF-19	Aprobar adopción	8	8	8	13	1,85	
modulo inteligente	RF-20	Sugerir animales según el perfil del interesado en la adopción.	3	5	13	13	1,62	1,62
donación en línea	RF-21	Realizar donación en línea.	5	8	13	21	1,24	1,24
modulo apadrinar	RF-22	Apadrinar una mascota.	2	5	5	13	0,92	0,82
	RF-23	Consultar apadrinamientos.	2	3	5	13	0,77	
	RF-24	Revocar apadrinamiento.	2	3	5	13	0,77	

Tabla 2 - WSJF

Módulo de gestión de usuarios

Se da una descripción detallada del funcionamiento del módulo de gestión de usuarios y se presenta cada una de las secciones del diagrama de casos de uso que se relacionan con el requerimiento en cuestión. Este módulo permite establecer los diferentes accesos e iteraciones del usuario con el sistema con el fin de crear y administrar los perfiles de los distintos usuarios de la aplicación.

ID:	RF-1	Nombre:	Registrar usuario		
Valor:	21	Prioridad:	1	Estimación:	20 horas
Descripción:	Como usuario deseo registrarme en el sistema para hacer uso de las funcionalidades disponibles.				
Criterio de validación:	Cuando los datos son correctos el sistema permitirá la creación del nuevo usuario. Los administradores pueden crear usuarios de tipo administrador.				

Tabla 3 Registrar usuario

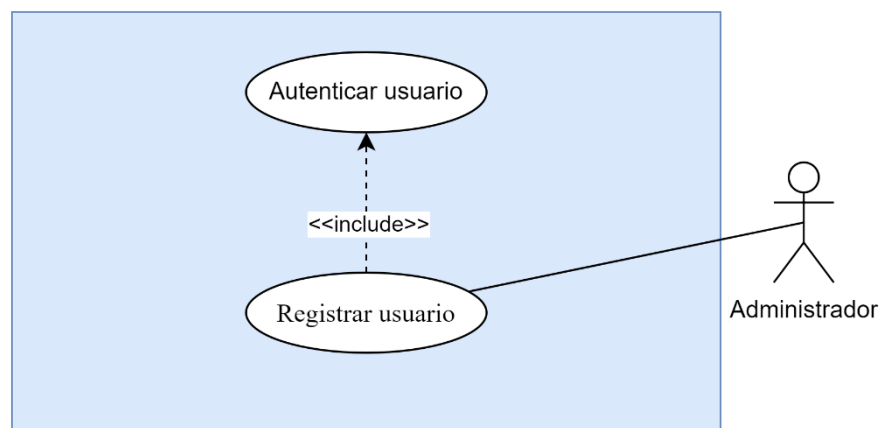


Ilustración 7 Caso de uso creación de usuarios

ID:	RF-2	Nombre:	Autenticar usuario.		
Valor:	21	Prioridad:	2	Estimación:	18 horas
Descripción:	Como usuario deseo validar mi identidad ante el sistema, para hacer uso de las funcionalidades disponibles.				

Criterio de validación:	Cuando el usuario (correo) y contraseña son correctos el sistema permite el ingreso. Los usuarios administradores del sistema ingresan al panel con sus respectivas funcionalidades. Cuando el usuario o contraseña son incorrectos el sistema muestra un mensaje diciendo: "ingrese de nuevo usuario y contraseña"
-------------------------	---

Tabla 4 RF-2 Autenticar usuario

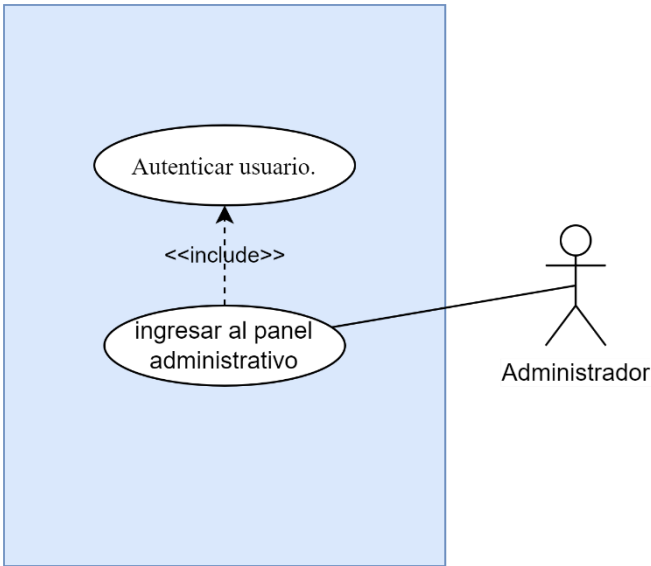


Ilustración 8 Caso de uso autenticar usuarios

ID:	RF-3	Nombre:	Modificar información del perfil de usuario.		
Valor:	21	Prioridad:	4	Estimación:	10
Descripción:	Como usuario deseo poder editar la información de mi perfil para mantenerla actualizada en todo momento.				
Criterio de validación:	Cuando los datos son correctos el sistema permite actualizar los datos del usuario. Los administradores pueden cargar imágenes o eliminar las que se encuentran en su perfil. Antes de actualizar los datos el sistema requiere que se confirme del cambio hecho.				

Tabla 5 RF-3 Modificar información del perfil de usuario.

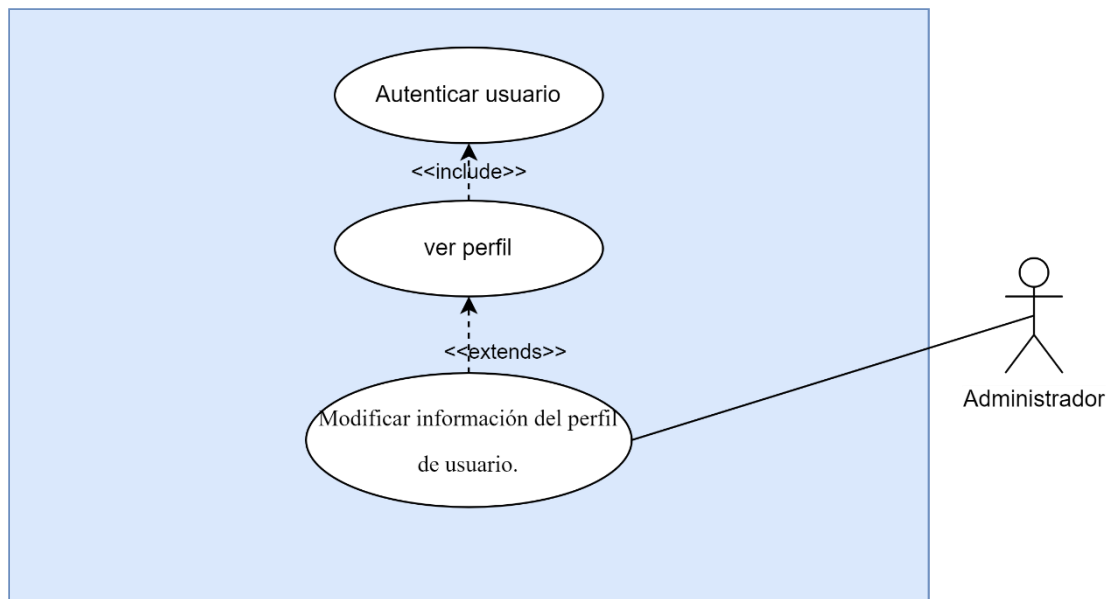


Ilustración 9 caso de uso modificar información del perfil de usuario

ID:	RF-4	Nombre:	Modificar contraseña del usuario.		
Valor:	21	Prioridad:	14	Estimación:	10
Descripción:	Como usuario deseo poder editar la contraseña de mi perfil para estar mayor seguridad en mi perfil.				
Criterio de validación:	Cuando los datos son correctos el sistema permite actualizar la contraseña del usuario. Antes de actualizar los datos el sistema requiere que se confirme del cambio hecho.				

Tabla 6 RF-4 Modificar contraseña del usuario.

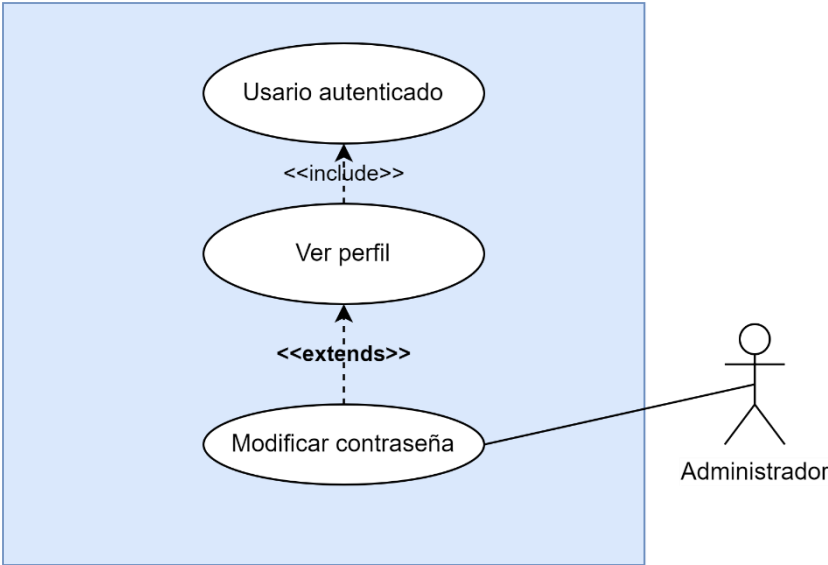


Ilustración 10 Caso de uso modificar contraseña del usuario

ID:	RF-5	Nombre:	Eliminar usuario.		
Valor:	2	Prioridad:	5	Estimación:	4 horas
Descripción:	Como usuario administrador deseo poder eliminar a otros usuarios para controlar el acceso al sistema. Evitando problemas de seguridad.				
Criterio de validación:	Cuando el administrador elimina a un usuario, éste ya no puede ingresar al sistema. Los datos del usuario no se eliminan para mantener la integridad de la base de datos.				

Tabla 7 RF-5 Eliminar usuario.

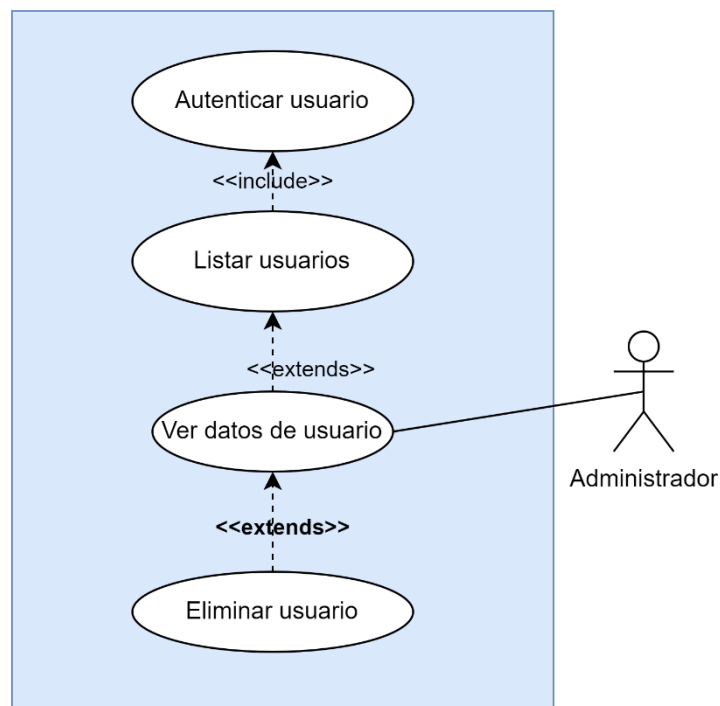


Ilustración 11 Caso de uso: Eliminar usuario

Módulo de gestión animal

Se da una descripción detallada del funcionamiento del módulo de gestión animal y se presenta cada una de las secciones del diagrama de casos de uso que se relacionan con el requerimiento en cuestión. Este módulo permite establecer los diferentes accesos e iteraciones del usuario con el sistema con el fin de crear y administrar los datos de los animales.

ID:	RF-6	Nombre:	CRUD animales		
Valor:	13	Prioridad:	8	Estimación:	20 horas
Descripción:	Como administrador, deseo registrar, editar, listar, ver y eliminar animales para garantizar información verídica y actualizada a los adoptantes.				
Criterio de validación:	Cuando los datos son correctos el sistema permite la creación de un nuevo animal. El sistema permite editar los datos del animal, incluyendo habilitar un animal para adopción. Antes de eliminar un animal el sistema pide confirmar la eliminación.				

Tabla 8 RF-6 CRUD animales

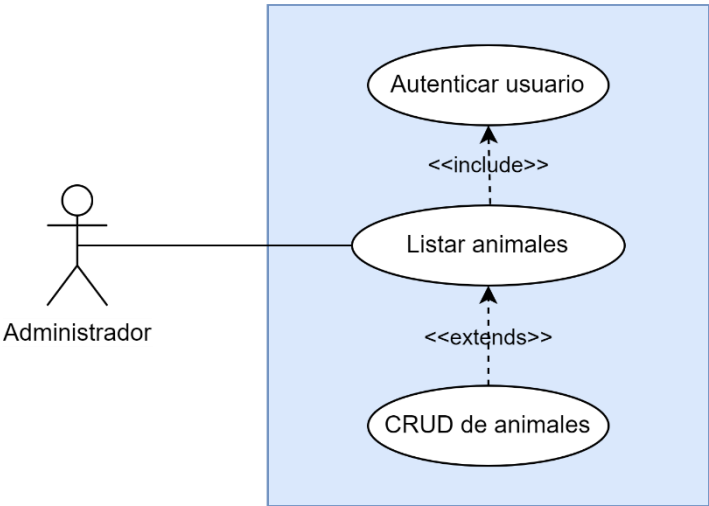


Ilustración 12 Caso de uso CRUD de animales

ID:	RF-7	Nombre:	Listar animales		
Valor:	3	Prioridad:	24	Estimación:	5 horas
Descripción:	Como listar animales para comprobar información verídica y actualizada.				
Criterio de validación:	El sistema permite consultar los animales que se encuentran registrados				

Tabla 9 RF-7 Listar animales.

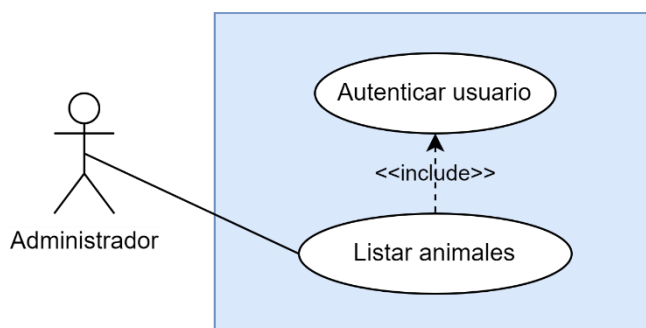


Ilustración 13 Caso de uso: Listar animales

ID:	RF-8	Nombre:	CRUD Razas		
Valor:	13	Prioridad:	8	Estimación:	20 horas
Descripción:	Como administrador, deseo registrar, editar, listar, ver razas para garantizar información verídica y actualizada a los adoptantes.				
Criterio de validación:	Cuando los datos son correctos el sistema permite la creación de una nueva raza.				

Tabla 10 RF-8 CRUD Razas

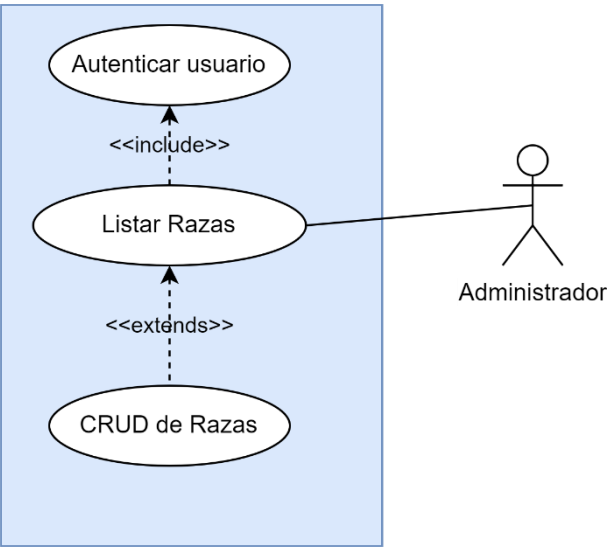


Ilustración 14 Caso de uso CRUD de Razas

Módulo de gestión de adopción

Se da una descripción detallada del funcionamiento del módulo de gestión de adopción y se presenta cada una de las secciones del diagrama de casos de uso que se relacionan con el requerimiento en cuestión. Este módulo permite establecer los diferentes accesos e iteraciones del usuario con el sistema con el fin de crear y administrar las adopciones de animales.

ID:	RF-10	Nombre:	Consultar animales para adopción.		
Valor:	13	Prioridad:	9	Estimación:	15 horas
Descripción:	Como adoptante, deseo consultar un listado de los animales con sus respectivos datos y fotos, que se encuentran disponibles para adoptar.				
Criterio de validación:	El sistema solo mostrará los animales que están disponibles para su adopción. Cuando un animal es adoptado ya no puede aparecer en esta lista.				

Tabla 11 RF-10 Consultar animales para adopción.

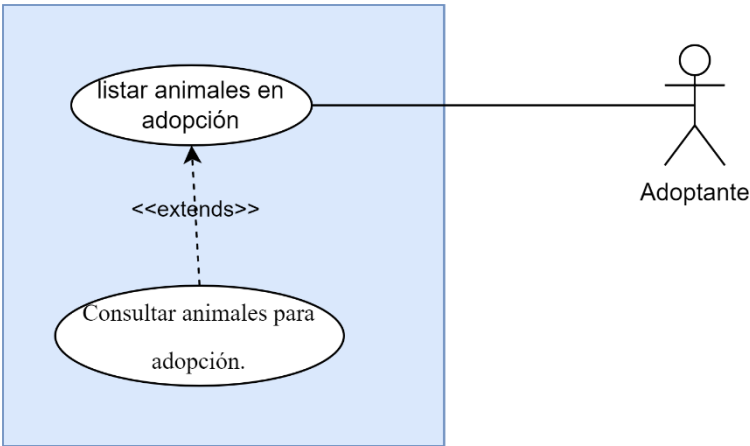


Ilustración 15 Caso de uso Consultar animales para adopción.

ID:	RF-11	Nombre:	Buscar animales		
Valor:	8	Prioridad:	17	Estimación:	15 horas
Descripción:	Como adoptante, deseo ingresar un perfil con mis preferencias a la hora de adoptar algún animal.				
Criterio de validación:	El sistema solo mostrara los animales según el criterio ingresado por el adoptante. Los filtros realizados por el criterio serán por: raza, tamaño, edad y color.				

Tabla 12 RF-11 Buscar animales

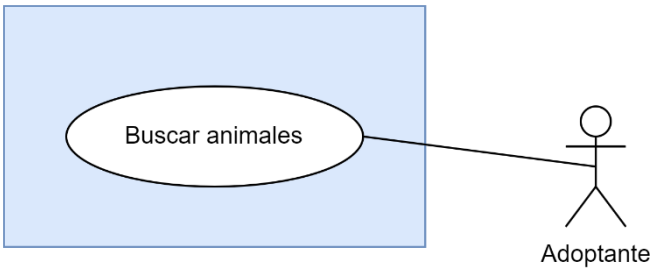


Ilustración 16 Caso de uso Buscar animales

ID:	RF-12	Nombre:	Enviar solicitud de adopción.		
Valor:	8	Prioridad:	6	Estimación:	8 horas
Descripción:	Como adoptante, deseo enviar la solicitud de adopción de un animal a la fundación que está a cargo de su cuidado, para su posterior evaluación y si es el caso su aprobación.				
Criterio de validación:	Una mascota puede recibir varias solicitudes de adopción por diferentes adoptantes, pero solo la fundación puede aprobar una solicitud. El adoptante puede realizar varias solicitudes de adopción a diferentes mascotas.				

Tabla 13 RF-12 Enviar solicitud de adopción.

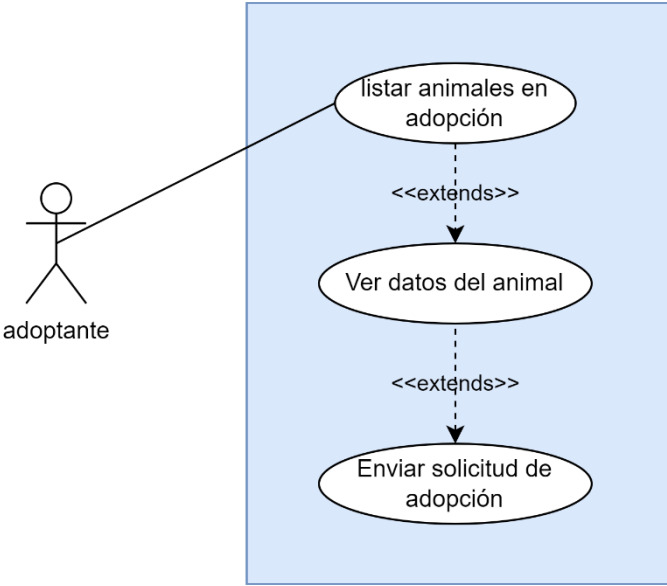


Ilustración 17 Caso de uso: Enviar solicitud de adopción.

ID:	RF-13	Nombre:	Cancelar solicitud de adopción		
Valor:	5	Prioridad:	12	Estimación:	4 horas
Descripción:	Como adoptante, deseo cancelar la solicitud de adopción de un animal o cancelarla si es dado el caso.				
Criterio de validación:	El adoptante puede cancelar la solicitud de adopción en cualquier momento.				

Tabla 14 RF-14 Modificar la solicitud de adopción

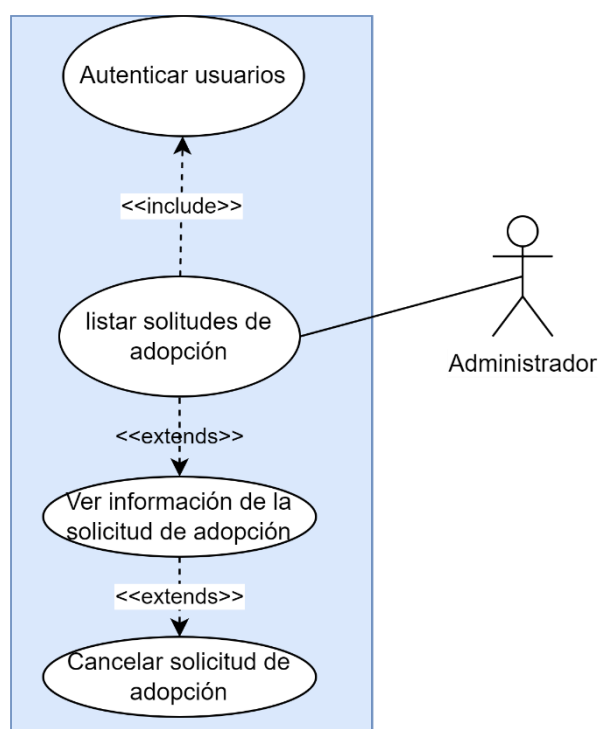


Ilustración 18 Caso de uso: Modificar la solicitud de adopción

ID:	RF-14	Nombre:	Listar solicitudes de adopción		
Valor:	5	Prioridad:	22	Estimación:	4 horas
Descripción:	Como administrador deseo ver las solicitudes de adopción que cada animal a mi cuidado con el fin de evaluar el adoptante más adecuado.				
Criterio de validación:	El administrador puede ver los animales que tienen solicitudes de adopción y para cada animal lista las solicitudes de adopción según lo requiera.				

Tabla 15 RF-15 Listar solicitudes de adopción

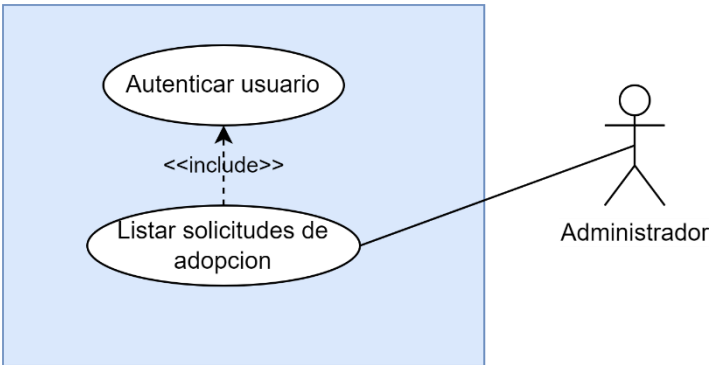


Ilustración 19 Caso de uso: Listar solicitudes de adopción

ID:	RF-15	Nombre:	Ver información de la solicitud de adopción		
Valor:	5	Prioridad:	10	Estimación:	4 horas
Descripción:	Como administrador deseo ver la información del animal que tiene la solicitud de adopción y la información del interesado (nombre, apellidos, teléfono y correo) de cada solicitud de adopción.				
Criterio de validación:	El administrador podrá ver la información (datos del adoptante) correspondiente a la solicitud de adopción seleccionada.				

Tabla 16 RF-16 Ver información de la solicitud de adopción

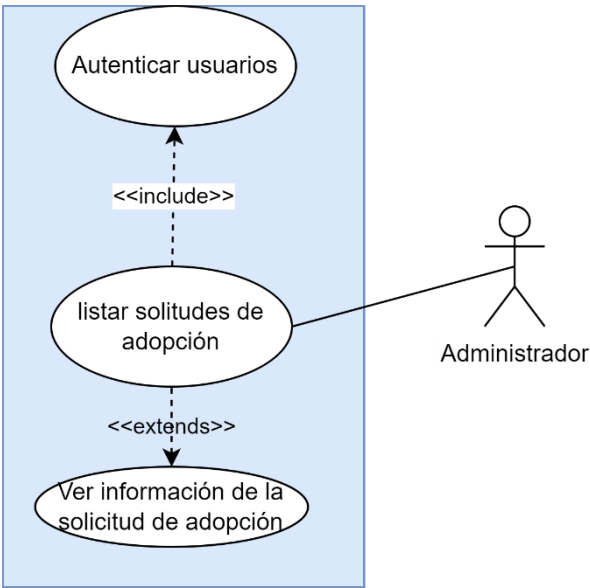


Ilustración 20 Caso de uso: Consultar datos de solicitud de adopción.

ID:	RF-16	Nombre:	Listar adopciones		
Valor:	1	Prioridad:	23	Estimación:	4 horas
Descripción:	Como administrador deseo listar las adopciones realizadas en la fundación.				
Criterio de validación:	El sistema debe permitir ver un listado de los datos de las adopciones que se han realizado en la fundación.				

Tabla 17 RF-18 Listar adopciones.

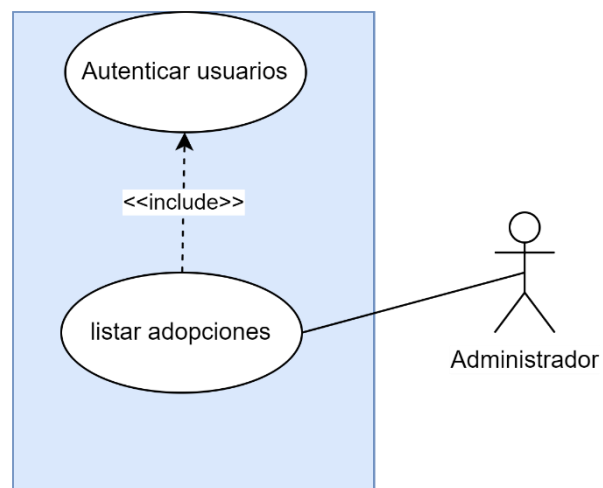


Ilustración 21 Caso de uso: Listar adopciones

ID:	RF-17	Nombre:	Aprobar Adopción		
Valor:	3	Prioridad:	24	Estimación:	5 horas
Descripción:	Como administrador deseo aprobar el estado de la solicitud de adopción según lo amerite la ocasión.				
Criterio de validación:	El proceso de adopción posee los siguientes estados: Activo: El animal está disponible para adopción. Inactivo: El animal ya no se encuentra disponible para adopción. Cuando un animal es adoptado el proceso de adopción debe cambiar automáticamente a inactivo.				

Tabla 18 RF-19 Aprobar Adopción

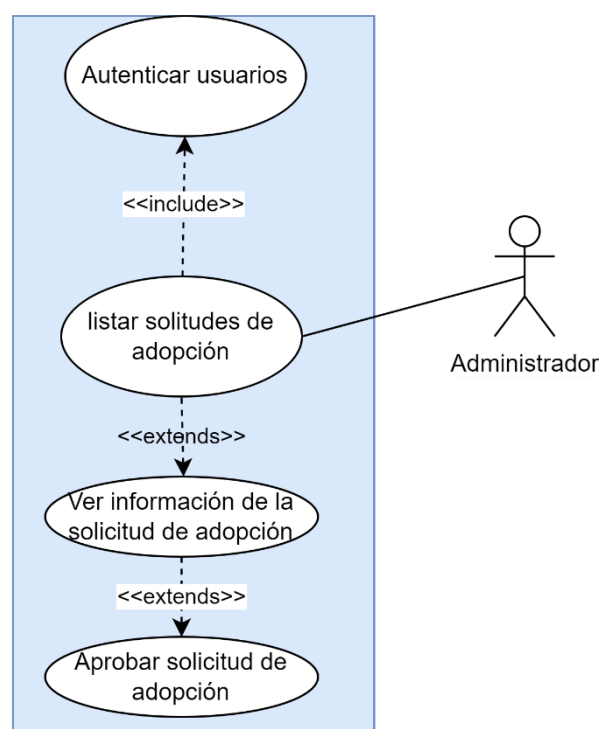


Ilustración 22 Caso de uso: Aprobar adopción

ID:	RF-18	Nombre:	Sugerir animales		
Valor:	3	Prioridad:	24	Estimación:	5 horas
Descripción:	Como adoptante deseo que los animales listados, disponibles para adopción se adapten a mis preferencias y gustos.				
Criterio de validación:	El sistema selecciona los animales según el perfil del adoptante y muestra según estos criterios.				

Tabla 19 RF-20 Sugerir animales.

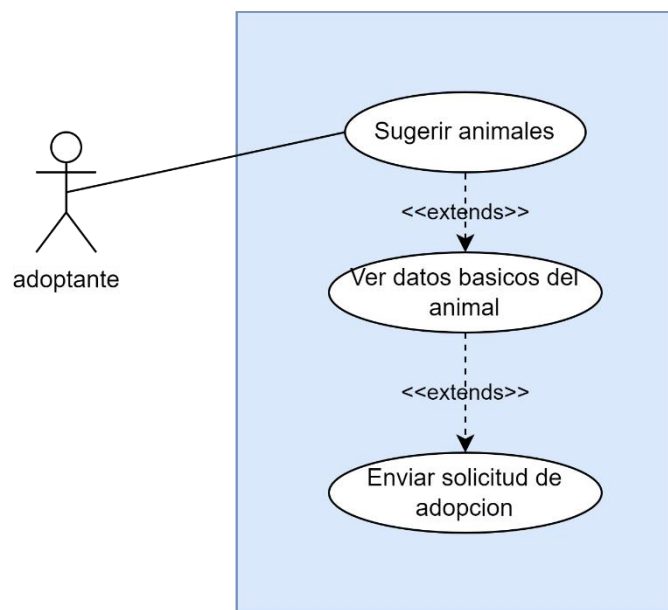


Ilustración 23 Caso de uso: Sugerir animales según el perfil del interesado en la adopción.

ID:	RF-19	Nombre:	Ver datos básicos del animal		
Valor:	5	Prioridad:	10	Estimación:	4 horas
Descripción:	Como adoptante deseo ver los datos básicos del animal como nombre, edad, color y raza.				
Criterio de validación:	El adoptante podrá ver los datos básicos del animal.				

Tabla 20- RF- 22 - Ver datos básicos del animal

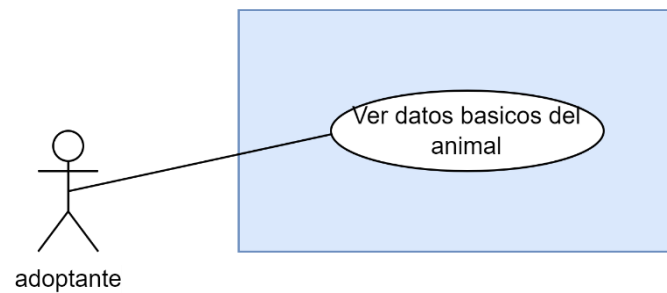


Ilustración 24- Caso de uso Ver datos básicos del animal

Módulo donación

Se da una descripción detallada del funcionamiento del módulo de donación y se presenta cada una de las secciones del diagrama de casos de uso que se relacionan con el requerimiento en cuestión. Este módulo permite establecer los diferentes accesos e iteraciones del usuario con el sistema con el fin de realizar donaciones en línea

ID:	RF-20	Nombre:	Realizar donación en línea.		
Valor:	5	Prioridad:	21	Estimación:	15 horas
Descripción:	Como usuario del sistema me gustaría realizar donaciones en línea de esta forma apoyar el cuidado de los animales en situación de calle.				
Criterio de validación:	El pago realizado en línea se realiza por mercado pago. Y las donaciones van una cuenta de la fundación.				

Tabla 21 RF-21 Realizar donación en línea.

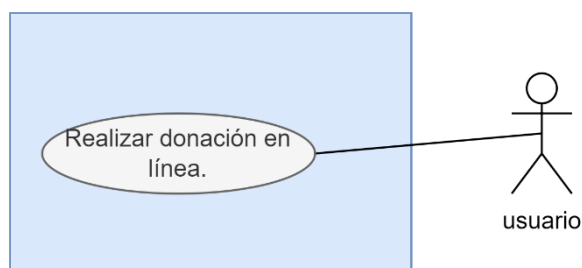


Ilustración 25 Caso de uso: Realizar donación en línea.

ID:	RF-21	Nombre:	Registrar apadrinamiento		
Valor:	5	Prioridad:	21	Estimación:	15 horas
Descripción:	Como usuario del sistema me gustaría registrarme como padrino de esta forma apoyar el cuidado de los animales en la fundación.				

Criterio de validación:	Se realiza el registro de los datos del apadrinamiento, nombre, teléfono, correo y dirección. De esta forma la fundación envía información.
-------------------------	---

Tabla 22 RF-21 Realizar donación en línea.

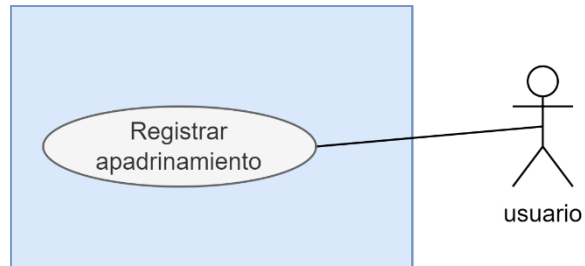


Ilustración 26 Caso de uso: Registrar apadrinamiento.

Diagrama de caso de uso

En el siguiente Ilustración 27 Diagrama caso de uso general se puede apreciar una visión más general del caso de como los actores interactúan con las diferentes partes del sistema.

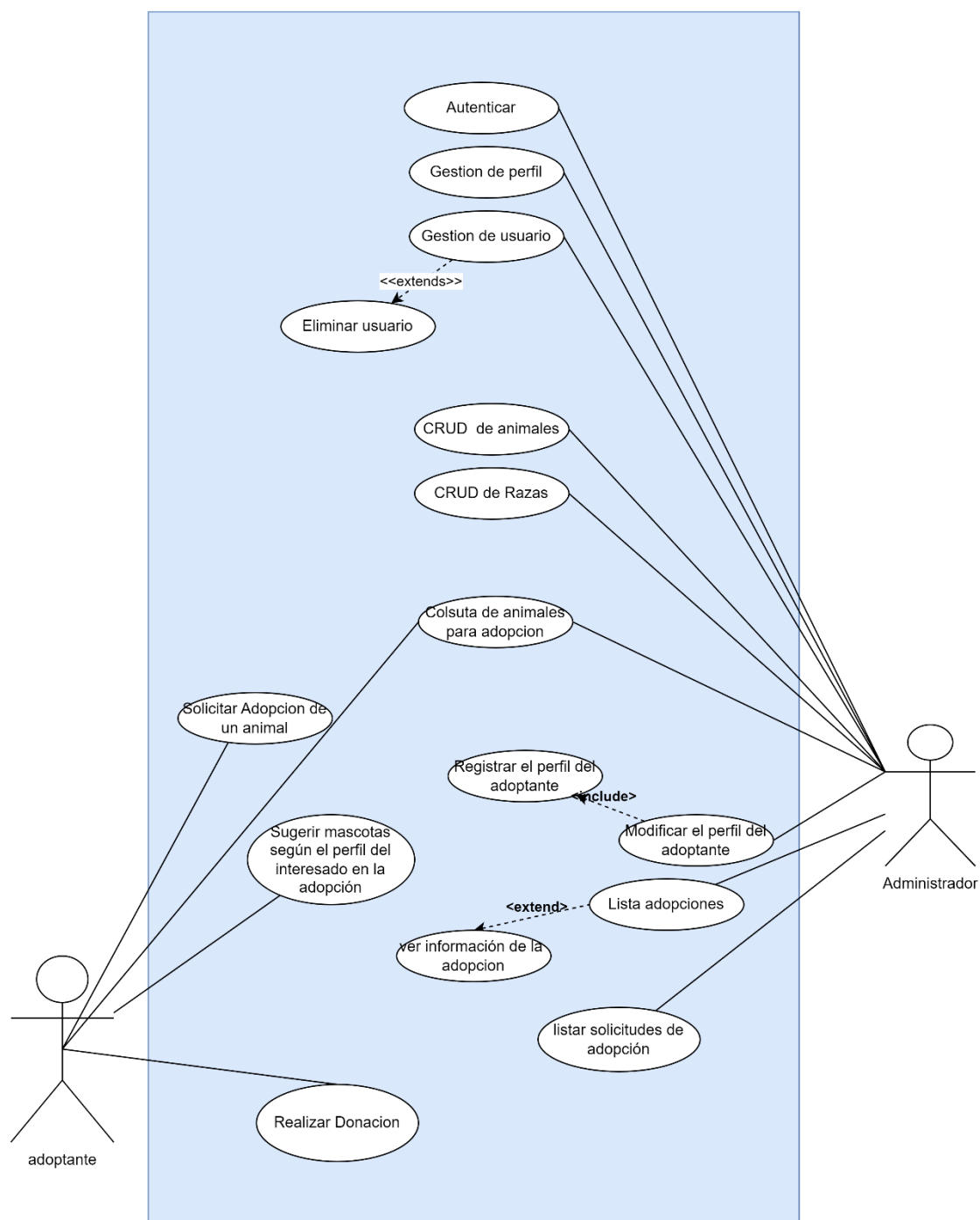


Ilustración 27 Diagrama caso de uso general

Requerimientos no funcionales

Estos requisitos son propiedades que debe tener el sistema, además imponen restricciones o requisitos a la aplicación, tanto en términos de apariencia como de requisitos para el funcionamiento del sistema. Se pueden distinguir de los comportamientos específicos del sistema enumerados en los requisitos funcionales. La aplicación móvil tiene los siguientes requisitos no funcionales para las propiedades de la aplicación y la experiencia del usuario.

ID:	RNF-1	Nombre:	Seguridad
Descripción:	Se debe permitir el ingreso al panel administrativo		
Criterio de validación:	Solo los usuarios autenticados correctamente pueden ingresar en el panel administrativo		

Tabla 23 RNF-1 Roles

ID:	RNF-2	Nombre:	Tiempo de Repuesta
Descripción:	Se deben manejar tiempos de respuesta que permitan la fluidez en el sistema		
Criterio de validación:	la repuesta a las peticiones que haga el usuario en el sistema deben ser menores a 3 segundos.		

Tabla 24 RNF-2 Tiempo de Repuesta

ID:	RNF-3	Nombre:	Amigable con el usuario
Descripción:	El sistema debe ser intuitivo y fácil de usar por los usuarios con poca experiencia.		
Criterio de validación:	El sistema debe ser usado correctamente con pocas o nulas indicaciones.		

Tabla 25 RNF-3 Amigable con el usuario

ID:	RNF-4	Nombre:	Datos personales
-----	-------	---------	------------------

Descripción:	El sistema debe garantizar que los datos proporcionados se mantienen solo para el propósito de las adopciones.
Criterio de validación:	Mantener los datos almacenados solo disponibles para las funciones específicas dentro de la aplicación

Diseño del Sistema

Diagrama de dominio

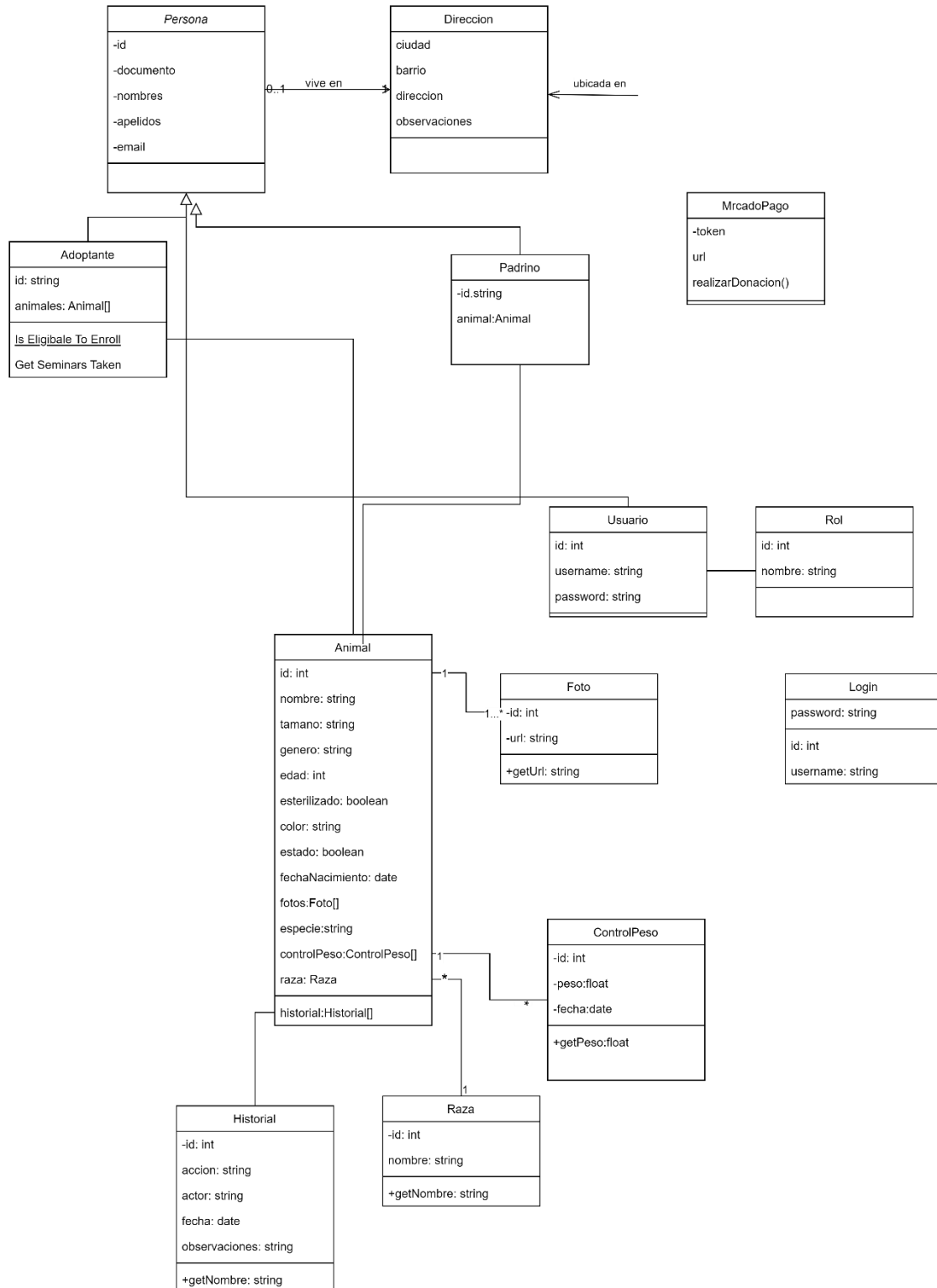


Ilustración 28 Diagrama Dominio

Diseño de la base de datos

El correcto diseño de una base de datos permite obtener acceso a la información exacta y actualizada, puesto que su diseño correcto nos permite cumplir con los demás procesos es por eso por lo que es necesario dedicar tiempo y esfuerzo en su creación.

Aunque la base de datos a utiliza es mongoDB que es una base de datos NoSQL lo cual quiere decir que la base de datos no almacena tablas ni relación con ellas, lo que la vuelve más flexible al momento de almacenar información en sus registros, se inicia el diseño realizando diagrama entidad relación

Diagrama Entidad Relación

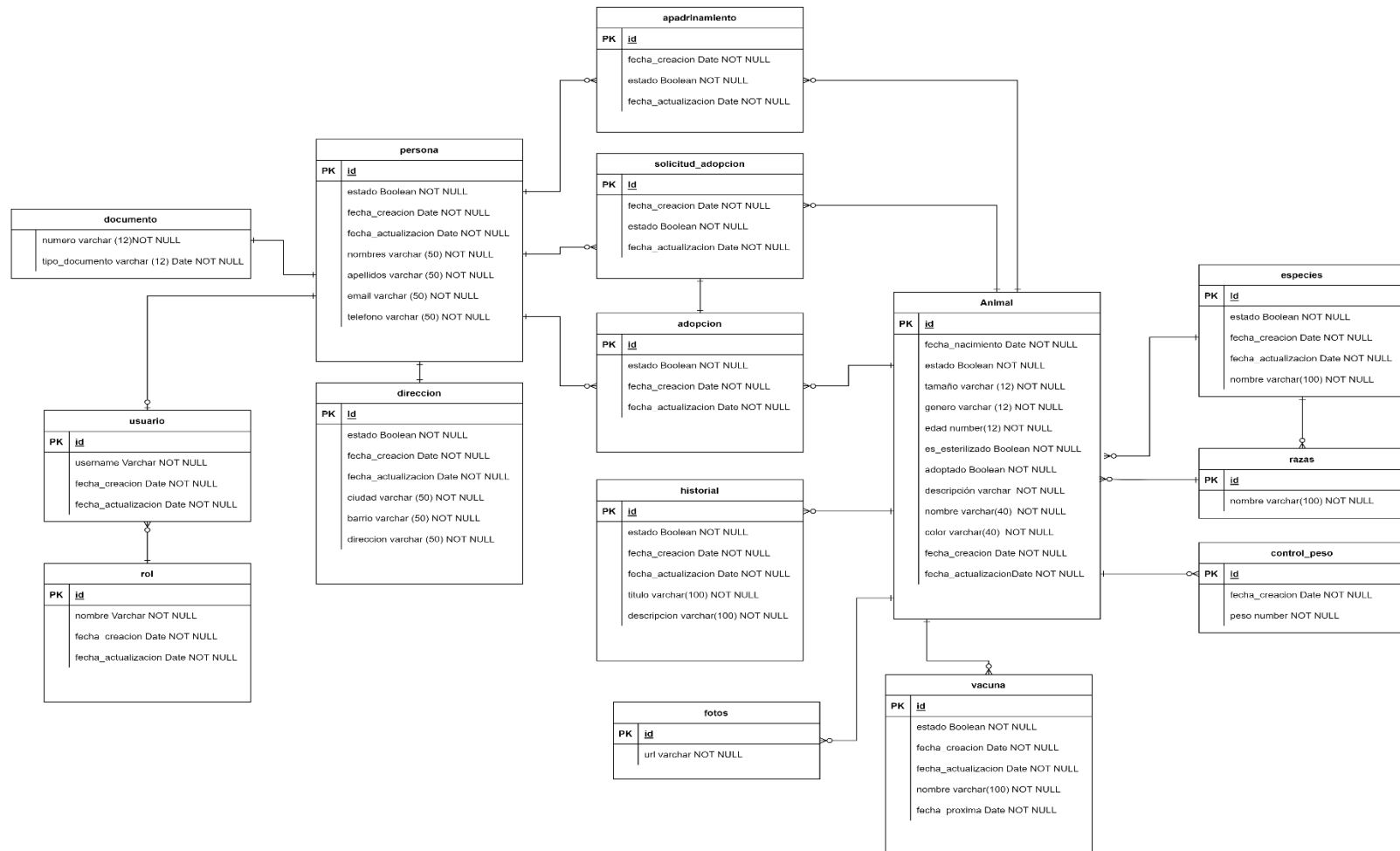


Ilustración 29 Diagrama Entidad Relación

El diagrama entidad relación (ver Ilustración 29 Diagrama Entidad Relación) es solo una representación que sirvió como guía en la construcción del esquema de datos que serán almacenados en MongoDB el cual es un SGBD documental y está compuesto por colecciones. Como lo muestra en la siguiente imagen:

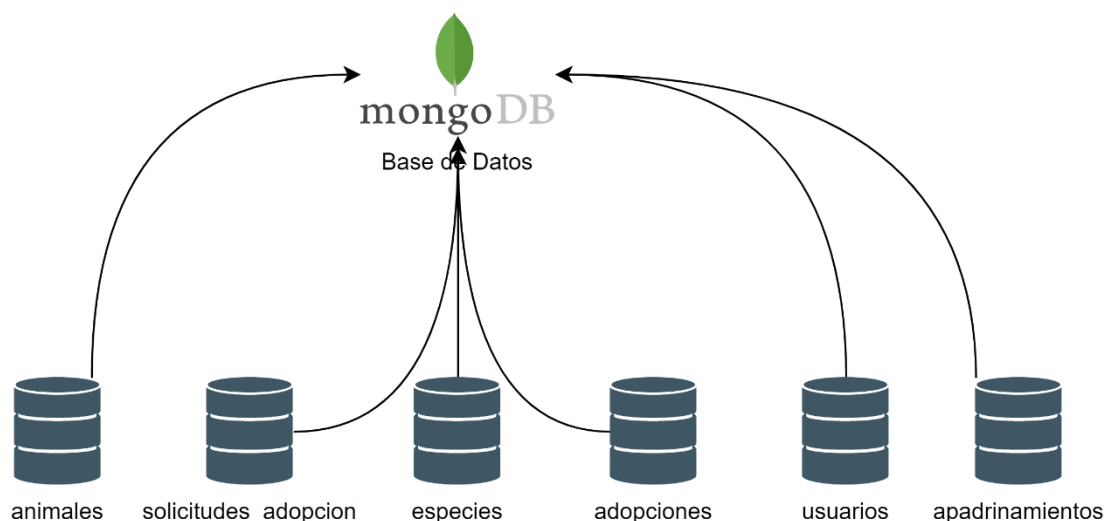


Ilustración 30 Diagrama Colecciones en MongoDB

Modelo Documental

A continuación, se describen algunas colecciones que hacen parte de la base de datos (ver Ilustración 30 Diagrama Colecciones en MongoDB).

Colección usuarios		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único
username	alfanumérico	nombre del usuario
email	alfanumérico	Correo electrónico del usuario
password	alfanumérico	Contraseña o clave de acceso.
rol	objeto	Objeto que contiene el nombre del rol del usuario
fullname	alfanumérico	Nombre completo del usuario
image_perfil	alfanumérico	Url de la imagen del perfil del usuario

Tabla 26-Diccionario de Datos Colección Usuarios

La Tabla 26-Diccionario de Datos Colección Usuarios define la estructura y característica lógica para esta colección en la Base de Datos, de esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 31 Documento usuario).

```
{
  "_id": {
    "$oid": "60724bd46c347501c23369ca"
  },
  "username": "admin",
  "email": "blade_liger13@hotmail.com",
  "password": "cd5ea73cd58f827fa78eef7197b8ee606c99b2e6",
  "rol": {
    "name": "admin"
  },
  "fullname": "Administrador",
  "image_perfil": "https://i1.sndcdn.com/avatars-000587714706-vjdrog-t200x200.jpg"
}
```

Ilustración 31 Documento usuario

Colección Animal		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único
nombre	alfanumérico	nombre del animal
color	alfanumérico	Color del animal
tamaño	alfanumérico	Tamaño del animal (pequeño, mediano, grande)
esterilizado	booleano	Indicador de si el animal esta esterilizado
en_adopcion	booleano	Indicador de si el animal se encuentra en adopción
estado	booleano	Indicador de si el animal se encuentra disponible
descripcion	alfanumérico	Descripción u observaciones de los animales que ayudan en su identificación
fecha_nacimiento	fecha	Fecha de nacimiento del animal
fotos	Array	lista de objetos que contienen las url de las fotos de los animales
especie	alfanumérico	Indica la clasificación animal (perro, gato)
raza	alfanumérico	Indica la raza del animal
sexo	alfanumérico	Indica el sexo del animal F para hembras y M para machos

score	numérico	Permite agrega puntos a los animales para cambiar el orden en el que son presentados
vacunas	Array	Lista de objetos que contiene la información de las vacunas
control_peso	Array	Lista de objetos que contiene la información de los pesos del animal

Tabla 27- Diccionario de Datos Colección Animal

La Tabla 27- Diccionario de Datos Colección Animal define la estructura y característica lógica para esta colección en la Base de Datos, de esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 32 Documento animal).

```
{
  "_id":{ },
  "nombre":"docky",
  "color":"negro",
  "tamaño":"pequeño",
  "esterilizado":false,
  "en_adopcion":false,
  "estado":true,
  "descripcion":"madre de 3 cachorrrros",
  "fecha_nacimiento":{ },
  "fotos":[
    {
      "id":1,
      "url":"https://picsum.photos/id/237/200/300"
    }
  ],
  "especie":"perro",
  "raza":"criollo",
  "vacunas":[ ],
  "control_peso":{ },
  "sexo":"F"
}
```

Ilustración 32 Documento animal

Colección Especies		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único
nombre	alfanumérico	nombre de la especie
razas	Array	Lista de las razas que puede tener una especie
createdAt	fecha	Fecha de la creación de la especie
updatedAt	fecha	Fecha de actualización de información de la especie

Tabla 28- Diccionario de Datos Colección Especies

La Tabla 28- Diccionario de Datos Colección Especies define la estructura y característica lógica para esta colección en la Base de Datos, de esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 33-Documento Especies).

```
{
  "_id": {
    "$oid": "607221576c347501c23369b9"
  },
  "nombre": "Gato",
  "razas": [{
    "id": 0,
    "nombre": "Criollo"
  }, {
    "id": 1,
    "nombre": "Siames"
  }]
}
```

Ilustración 33-Documento Especies

Colección solicitudes de adopción		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único relacionado
Adoptantes	Array	Lista de los objetos adoptantes interesados en un animal
createdAt	fecha	Fecha de la creación de la adopción
updatedAt	fecha	Fecha de actualización de información de la adopción

Tabla 29- Diccionario de Datos Colección solicitudes de adopción

La Tabla 29- Diccionario de Datos Colección solicitudes de adopción define la estructura y característica lógica para esta colección en la Base de Datos, dentro de esta se almacenan las solicitudes de adopción de varias personas con cada mascota. De esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 34-Documento solicitudes de adopciones).

```

1  {
2    "_id": {
3      "$oid": "635f1c17a4a0af5c82849b4e"
4    },
5    "adoptante": [
6      {
7        "documento": {
8          "tipo": "CC",
9          "numero": "4546546"
10       },
11       "nombres": "milena",
12       "apellidos": "rojas",
13       "direccion": {
14         "ciudad": "cucuta",
15         "barrio": "bocono",
16         "direccion": "cr 7 #5-6"
17       },
18       "telefono": "565656",
19       "email": "blade_liger@hotmail.com",
20       "estado": "inicial"

```

Ilustración 34-Documento solicitudes de adopciones

Documento Adoptante		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único
documento	objeto	Objeto que contiene el tipo de documento y el número de documento
nombres	alfanumérico	Contiene los nombres del adoptante
apellidos	alfanumérico	Contiene los apellidos del adoptante
direccion	objeto	contiene la información de la dirección del adoptante (Ciudad, barrio, dirección)
telefono	alfanumérico	Teléfono del adoptante
email	alfanumérico	Correo electrónico del adoptante
estado	objeto	Contiene la información del estado de la adopción (inicial, en estudio, aprobado, pendiente, rechazado)
createdAt	fecha	Fecha de la creación de la adopción
updatedAt	fecha	Fecha de actualización de información de la adopción

Tabla 30- Diccionario de Datos Documento Adoptante

La Tabla 30- Diccionario de Datos Documento Adoptante define la estructura y característica lógica para esta colección en la Base de Datos, de esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 35 Documento adoptante).

```

{
  "_id": {
    "$oid": "607249da6c347501c23369c9"
  },
  "documento": {
    "tipo": "CC",
    "numero": "12345678"
  },
  "nombres": "sol",
  "apellidos": "correa",
  "direccion": {
    "ciudad": "cucuta",
    "barrio": "San martin",
    "direccion": "calle 5 #7-50"
  },
  "telefono": "3232331",
  "email": "example@email.com"
}

```

Ilustración 35 Documento adoptante

Colección apadrinamientos		
Nombre	Tipo de dato	Descripción
_id	alfanumérico	Identificador único
nombre	alfanumérico	Contiene el nombre de la fundación
direccion	objeto	Contiene la información de la dirección de la fundación (Ciudad, barrio, dirección)
telefono	alfanumérico	Teléfono de la fundación
email	alfanumérico	Correo electrónico de la fundación
createdAt	fecha	Fecha de la creación de la fundación
updatedAt	fecha	Fecha de actualización de información de la fundación

Tabla 31-Diccionario de datos Colección apadrinamientos

La Tabla 31-Diccionario de datos Colección define la estructura y característica lógica para esta colección en la Base de Datos, de esta forma se garantizan que se envíen datos esperados como son los del ejemplo (ver Ilustración 36 Documento fundación).

```

1 { "id": "60723cdf6c347501c23369c8",
2   "nombre": "patitas",
3   "direccion": {
4     "ciudad": "cucuta",
5     "barrio": "San martin",
6     "direccion": "calle 5 #7-50"
7   },
8   "telefono": "3213132313",
9 }

```

Ilustración 36 Documento fundación

Implementación de la base de datos

Para la implementación de la base de datos, utiliza el servicio de mongoDB Server para un ambiente de pruebas local. El despliegue de la base de datos se realiza en un servidor externo llamado MongoDB ATLAS, que es un Daas (Base de datos como servicio) donde ellos configuran y alojan la base de datos, librando al usuario de esas gestiones, lo único que debe hacer el usuario es suministrar a la base de datos información.

Para iniciar, en la página MongoDB ATLAS, debe crearse una cuenta y un nuevo proyecto con el fin de alojar la base de datos de la aplicación (ver Ilustración 37- Crear proyecto mongoDB).

Create a Project

Name Your Project Add Members

Name Your Project

Project names have to be unique within the organization (and other restrictions).

server-test

Cancel

Next

Ilustración 37-Crear proyecto mongoDB

Luego de la creación del proyecto se debe crear el cluster que almacenara la base de datos con la que trabajara la aplicación ver Ilustración 38-Proyecto mongoDB.

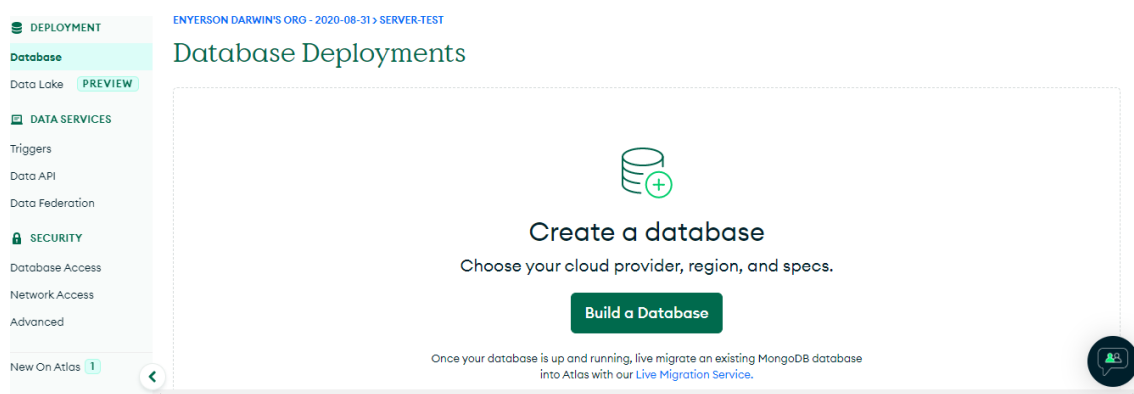


Ilustración 38-Proyecto mongoDB

La creación de clúster requiere escoger el tipo de servicio por el momento se seleccionará el servicio gratuito (ver Ilustración 39-Servicios mongoDB).

Continuando con el proceso corresponde seleccionar el proveedor y la región, se debe seleccionar un AWS que es un servicio que tiene disponible servicio en la región de Suramérica lo que dará mayor velocidad de acceso a la base de datos a los usuarios de que se encuentran en Cúcuta, Norte de Santander, Colombia.

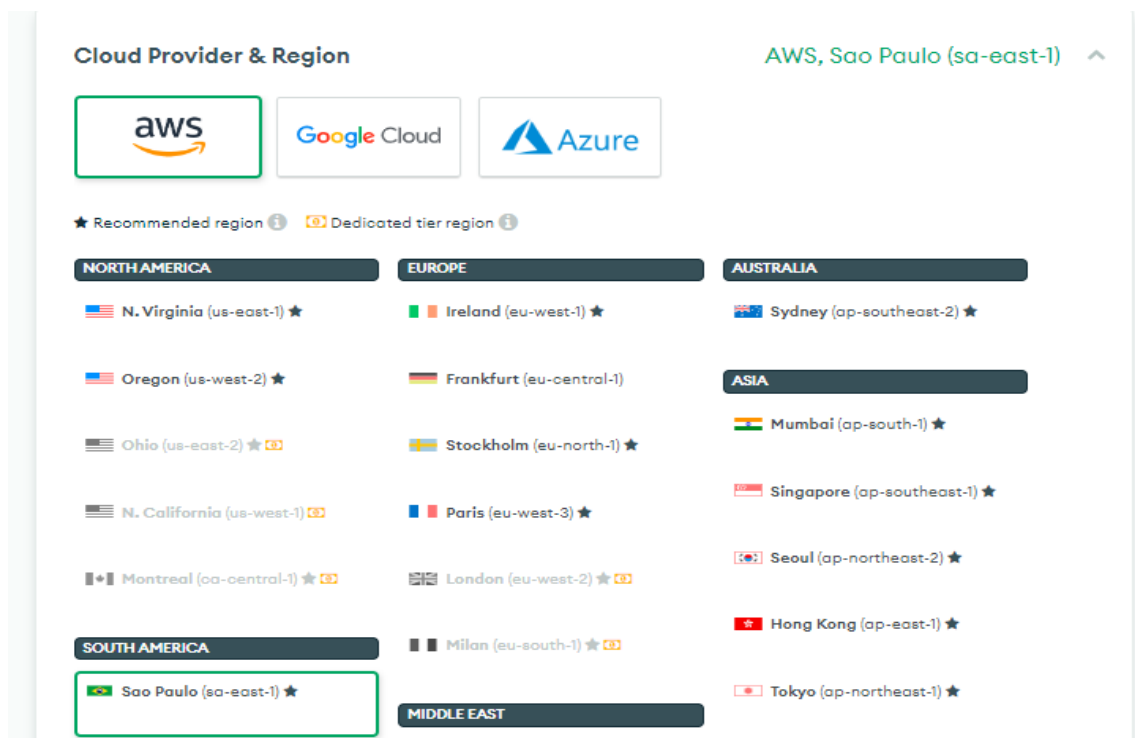


Ilustración 40-Proveedor y región mongoDB

Luego de seleccionar el proveedor y la región se crea el nuevo clúster, que solicitará un usuario con su respectiva contraseña para permitirle el acceso.

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username

Password

Ilustración 41-Creación de usuario mongoDB

A continuación, se crea el clúster y aparecerá el panel de administración

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SEARCH
5.0.14	AWS / Sao Paulo (so-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

Ilustración 42-Clúster mongoDB

Agregar la base de datos y la colecciones el paso a seguir para tener todo funcionando en el botón de Browse Collections de la Ilustración 42-Clúster mongoDB permitirá abrir una nueva interfaz para la creación de la base de datos y las colecciones.

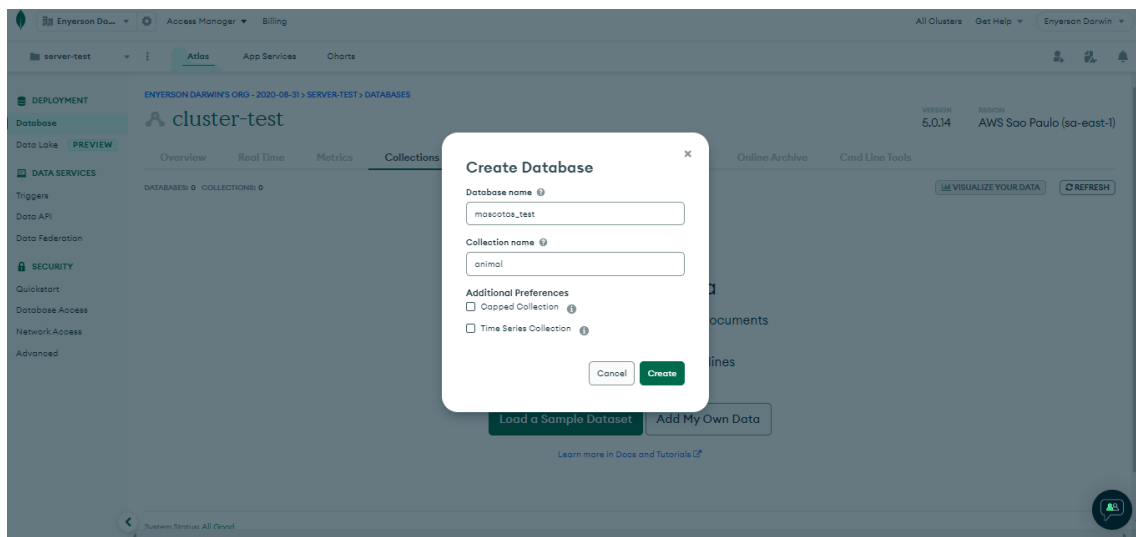


Ilustración 43-Creación Base de Datos MongoDB

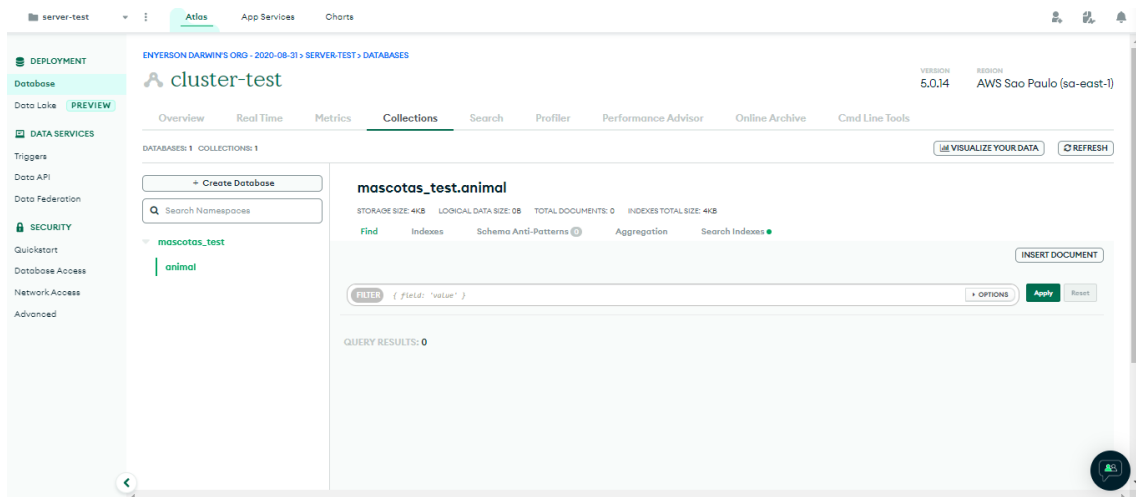


Ilustración 44-Base de Datos mongoDB

Diseño Arquitectónico Apis REST

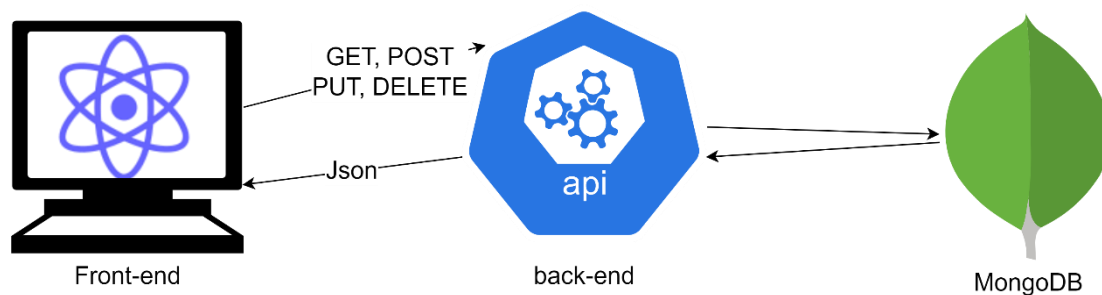


Ilustración 45-Diseño Arquitectura de Apis REST

Para la construcción de la aplicación, se diseñó con arquitectura de API REST , que este compuesto por parte ejecutada en el cliente (Front-end) y otro de lado del servidor (back-end) todas las solicitudes se manejan por verbos HTTP (GET, POST, PUT, DELETE) los cuales son atendidos por el back-end el cual responde con el código según corresponda 200 (OK), 204(Created), 401(Unauthorized) y la información devuelta se encuentra en formato JSON, que es un estándar y con la cual el front-end puede manejarla y mostrarla al usuario.

Para la comunicación entre el front-end, back-end y la base de datos se utiliza archivos de variables de entornos que permite cambiar el host donde está funcionando el servicio (back-end o base de datos) de esta forma se puede trabajar en un ambiente de desarrollo con una base de datos de desarrollo y sin afectar al ambiente de producción.

```
1 DATABASE =mascotas_db
2 DATABASE_URL = mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
3 SECRETKEY =Unipamplona
4 ENVIRONMENT_TYPE=T You, now * Uncommitted changes
```

Ilustración 46-archivo variables de entorno back-end

Como se puede observar en la ilustración en este archivo se especifica la Url de la base de datos en este caso apunta a un servicio local. Por otro lado, se puede observar el nombre de la base de datos la cual puede ser cambiada en diferentes ambientes para mantener la seguridad de los datos.

```
1 REACT_APP_API_URL=http://localhost:8080/api/  
2 REACT_APP_NAME=PATITAS APP
```

Ilustración 47-archivo variables de entorno front-end

En la Ilustración 47-archivo variables de entorno front-end se puede ver el archivo de variables de entorno del front-end el cual se le debe indicar la Url del api en la variable REACT_APP_API_URL la cual está apuntando a un servicio levantado localmente para apuntar a otro servicio basta con cambiar el valor de la variable de entorno.

Estructura de la aplicación

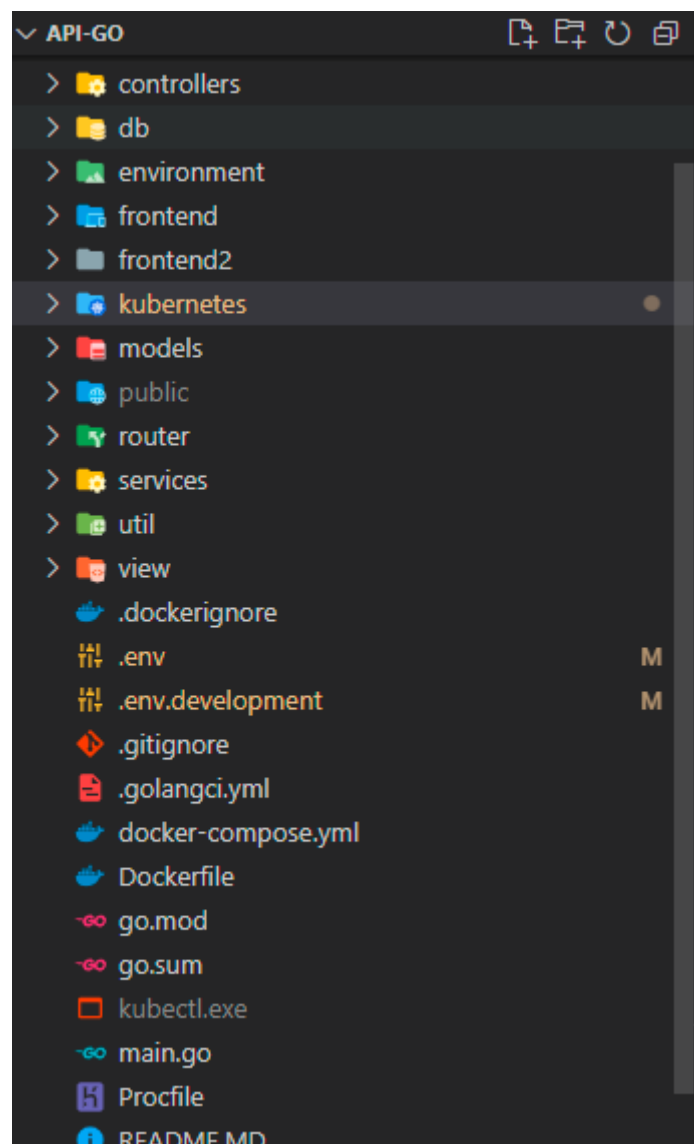


Ilustración 48-Estructura de la aplicación

Para la construcción de la aplicación se realiza utilizando la herramienta Visual Studio Code, se crean los directorios para organizar el proyecto donde el directorio de models contiene los structs que para Golang son homólogos de las clases en otros lenguajes de programación. En el directorio services se encuentra la lógica de la aplicación donde las conexiones de la base de datos se le pasa como parámetro a los servicios para ser usados por ellos. En el directorio controladores se obtienen los parámetros necesarios para ser pasados a los servicios y llamar a cada uno.

La construcción de la aplicación para la adopción compatible y apadrinamiento de animales en condición de calle para la ciudad de Cúcuta que será usada por una fundación que atiende y refugia varios animales y que posteriormente da en adopción a personas con los recursos necesarios que garanticen la salud de los animales.

Para la construcción de la ampliación se utilizarán tecnologías como Visual Studio Code como IDE (Entorno de desarrollo), MongoDB como sistema de gestión de Base de Datos no relacional, Go o Golang para la construcción de los servicios Apis REST, del lado del cliente la aplicación que consume los servicios expuestos por el back-end es construida con la librería React.js y MUI que proporciona un diseño amigable para los usuarios finales gracias a material design.

En el directorio router se encuentra el archivo donde se definen los diferentes endpoints con los respectivos verbos HTTP y que controlador será llamado como se manejan en el framework Iris.

```
68  api := app.Party("/api")
69  api.Post("/login", authController.Login)
70  api.Get("/pets/page/{page:uint64}/base/{base:uint64}", petsController.GetAll)
71  api.Get("/pets/count", petsController.Count)
72  api.Get("/pets/{id:string}", petsController.GetByID)
73  api.Post("/pets/{id:string}/adopt", adoptController.Save)
74  api.Use(util.Verify())
75  api.Use(j.Serve)
76  api.Get("/refresh", authController.RefreshToken)
77
```

Ilustración 49-Asignación de endpoints

El archivo para iniciar la aplicación es main.go que contiene llama a los demás servicios para su correcto funcionamiento y ejecutando simplemente el comando go run main.go inicia el servicio back-end.

```

15 func main() {
16     app := iris.New()
17     environment.ReadEnv("")
18     port := os.Getenv("PORT")
19     if port == "" {
20         port = "8080" //localhost
21     }
22
23     database := db.GetDatabase()
24
25     cors := cors.New(cors.Options{
26         AllowedOrigins: []string{"*"}, // allows everything, use that to change the hosts.
27         AllowedMethods: []string{"GET", "POST", "PUT", "DELETE"},
28         AllowCredentials: true,
29         AllowedHeaders: []string{"*"},
30     })
31
32     myrouter.AddRutas(app, database)
33
34     // Listens and serves incoming http requests
35     // on http://localhost:8080.
36     app.Listen(":" + port)
37 }
38
39

```

Ilustración 50-Archivo main.go

Una de las formas que Golang administran las dependencias es a través de un archivo llamado `go.mod` que consiste en una serie de configuraciones para establecer las dependencias y las versiones de cada una herramienta usada, en la que se agregan las Url de los repositorios GitHub donde se encuentran las librerías. De esta forma se define su utilización en el proyecto, ver Ilustración 51-Dependencias en Golang.

```

1 module github.com/edcamero/api-go
2
3 go 1.13
4
5 Check for upgrades | Reset go.mod diagnostics | Upgrade transitive dependencies | Upgrade direct dependencies
6 require (
7     github.com/Joker/hpp v1.0.0 // indirect
8     github.com/google/uuid v1.3.0
9     github.com/iris-contrib/middleware/cors v0.0.0-20210110101738-6d0a4d799b5d
10    github.com/iris-contrib/middleware/jwt v0.0.0-20210110101738-6d0a4d799b5d
11    github.com/joho/godotenv v1.3.0
12    github.com/k0kubun/colorstring v0.0.0-20150214042306-9440f1994b88 // indirect
13    github.com/kataras/iris/v12 v12.2.0-alpha4
14    github.com/mattn/go-colorable v0.1.8 // indirect
15    github.com/smartystreets/goconvey v1.6.4 // indirect
16    github.com/yudai/pp v2.0.1+incompatible // indirect
17    go.mongodb.org/mongo-driver v1.7.4
18 )

```

Ilustración 51-Dependencias en Golang

En el directorio front-end de la aplicación se encuentran las carpetas y archivos de la librería React.js, que tiene como fin vincular esta herramienta a la arquitectura full stack que se pretende implementar en este proyecto, como se puede apreciar en la Ilustración 52-Directorio front-end

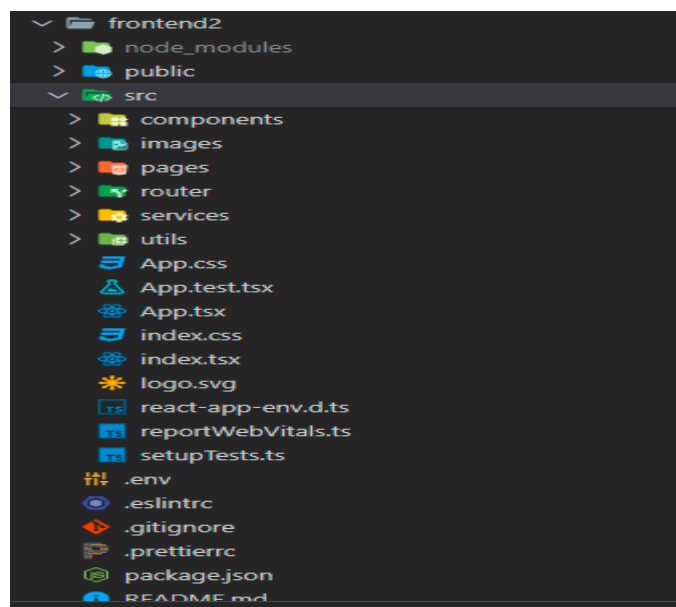
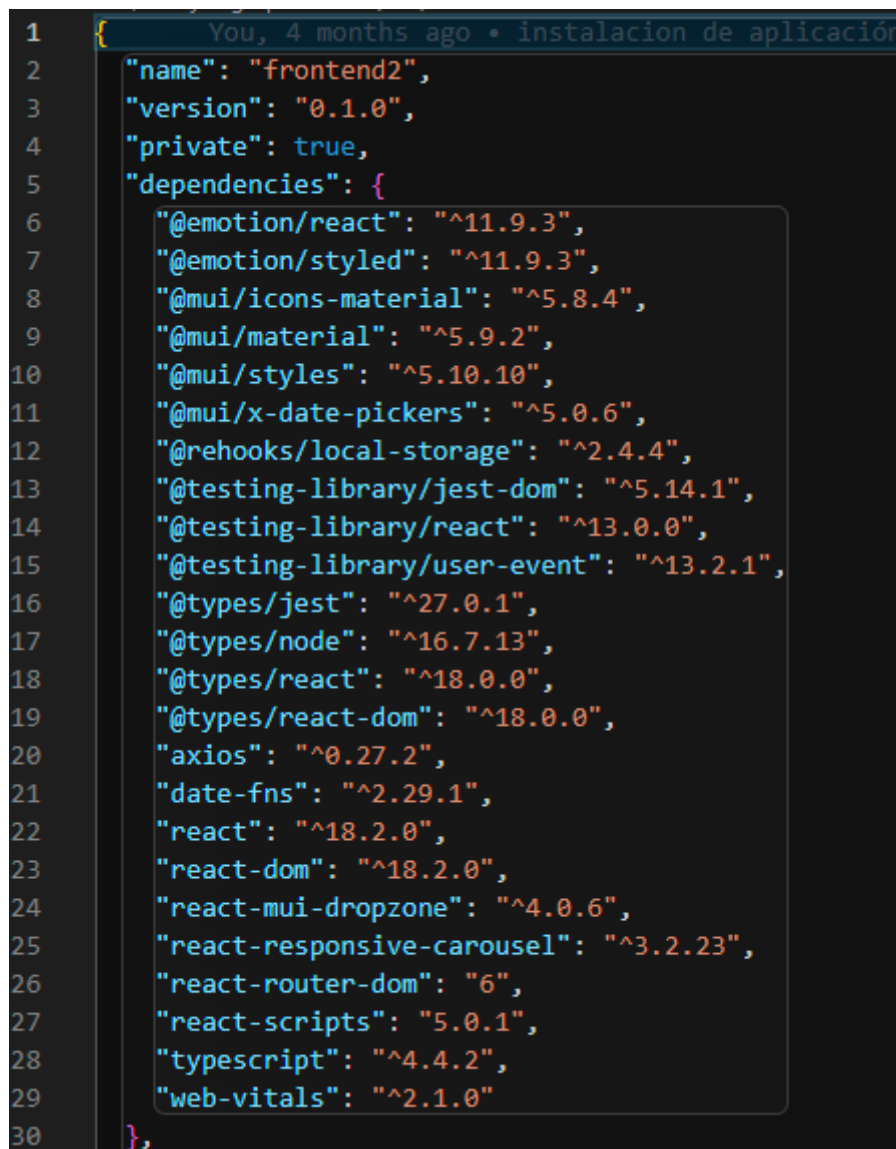


Ilustración 52-Directorio front-end

En la carpeta src del directorio front-end se encuentra toda la codificación necesaria para la presentación y vista grafica de la aplicación, entre los principales directorios se pueden encontrar el de los components, se utiliza para agregar los diferentes componentes que se pueden utilizar a lo largo de la aplicación. El directorio de pages se utiliza para almacenar las diferentes vistas o páginas en la que el usuario puede navegar. El directorio router se encuentra las rutas de acceso por las que el usuario puede navegar e ingresar a las diferentes páginas diseñadas. En el directorio services se tiene un archivo con el servicio que usa la librería axios, la cual es la librería con la que se realizan las peticiones a la API.

Las dependencias del front-end son administradas por el archivo package.json que contiene el nombre y versión de las librerías utilizadas. Ver Ilustración 53- Dependencias front-end



```

1  {
2    "name": "frontend2",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@emotion/react": "^11.9.3",
7      "@emotion/styled": "^11.9.3",
8      "@mui/icons-material": "^5.8.4",
9      "@mui/material": "^5.9.2",
10     "@mui/styles": "^5.10.10",
11     "@mui/x-date-pickers": "^5.0.6",
12     "@rehooks/local-storage": "^2.4.4",
13     "@testing-library/jest-dom": "^5.14.1",
14     "@testing-library/react": "^13.0.0",
15     "@testing-library/user-event": "^13.2.1",
16     "@types/jest": "^27.0.1",
17     "@types/node": "^16.7.13",
18     "@types/react": "^18.0.0",
19     "@types/react-dom": "^18.0.0",
20     "axios": "^0.27.2",
21     "date-fns": "^2.29.1",
22     "react": "^18.2.0",
23     "react-dom": "^18.2.0",
24     "react-mui-dropzone": "^4.0.6",
25     "react-responsive-carousel": "^3.2.23",
26     "react-router-dom": "6",
27     "react-scripts": "5.0.1",
28     "typescript": "^4.4.2",
29     "web-vitals": "^2.1.0"
30   },

```

Ilustración 53-Dependencias front-end

Desarrollo Front-end

Para el desarrollo del módulo se construyeron las vistas o interfaces que interactúan con el usuario, las cuales fueron diseñadas mediante la herramienta React.js que hace uso de archivos .tsx que permite el uso de JavaScript junto con HTML, a continuación, algunas de las pantallas que componen la aplicación.

Inicio

Al ingresar en la página en el usuario puede visualizar la página de inicio (ver Ilustración 54-Página de inicio)

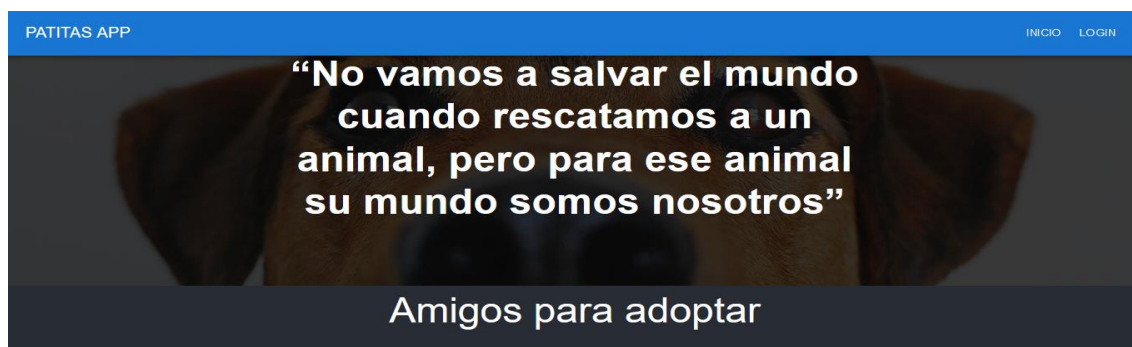


Ilustración 54-Página de inicio

Fuente: Elaboración propia

En esta misma página situado más abajo se puede encontrar los animales que se encuentran disponibles para su adopción (ver Ilustración 55-Animales en adopción). Solo se muestra información de 6 animales por cuestiones de rendimiento.

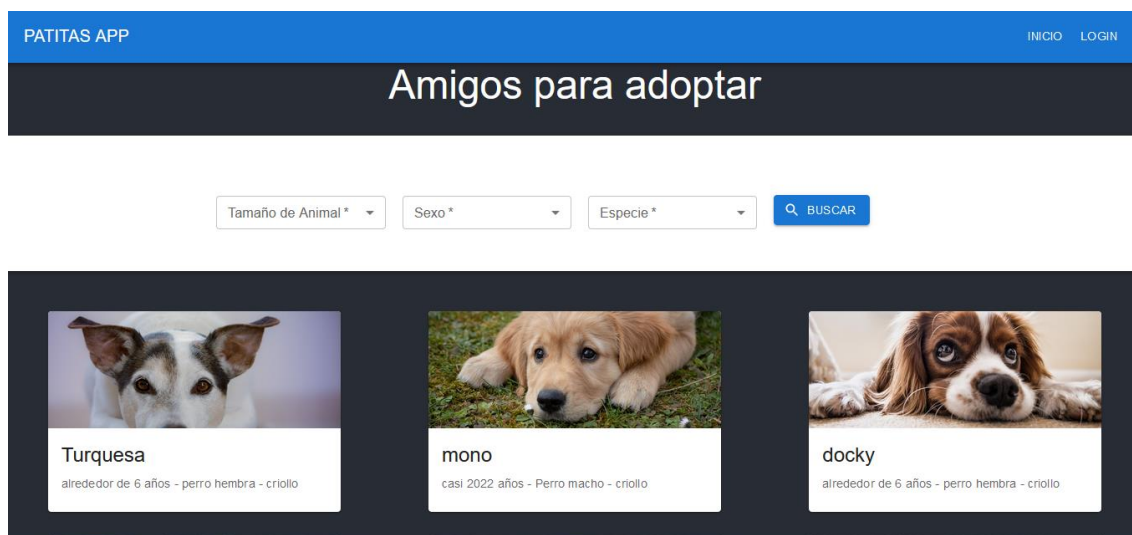


Ilustración 55-Animales en adopción

Fuente: Elaboración propia

Mostrar todos los animales disponibles en la base de datos puede causar problemas en la renderización de la información por lo que se agrega paginación en el

componente que muestra los animales disponibles (ver Ilustración 56-Paginación de animales disponibles)

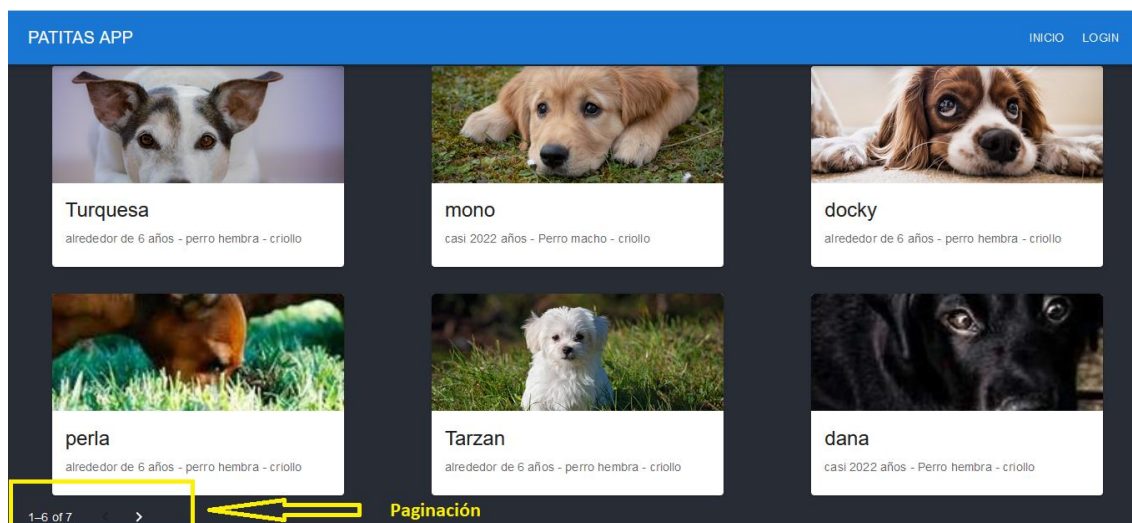


Ilustración 56-Paginación de animales disponibles

Fuente: Elaboración propia

En la parte de los animales disponibles para adoptar se encuentra un componente de tipo formulario que le permite al usuario seleccionar entre sus preferencias (tamaño, especie y sexo) los animales más acordes con sus gustos (ver Ilustración 55-Animales en adopción).

Continuando en la página de inicio, por debajo de los animales disponibles para adoptar, se encuentran los medios por los que las pueden contribuir con donaciones para la fundación (ver Ilustración 57-Medios de donación). Los dos primeros medios de pagos es solo información que el usuario debe captar manualmente para la realización de la donación. La opción de mercado pago es la que permite el pago en línea dentro de la aplicación y se mostrará más adelante.



Ilustración 57-Medios de donación

Fuente: Elaboración propia

Información del animal

En esta pantalla el usuario puede ver más información de su interés relacionada con el animal que ha seleccionado desde la página de inicio. Se puede conocer información como edad, si esta esterilizado, color, tamaño, sexo, especie y fotos que la fundación haya agregado.

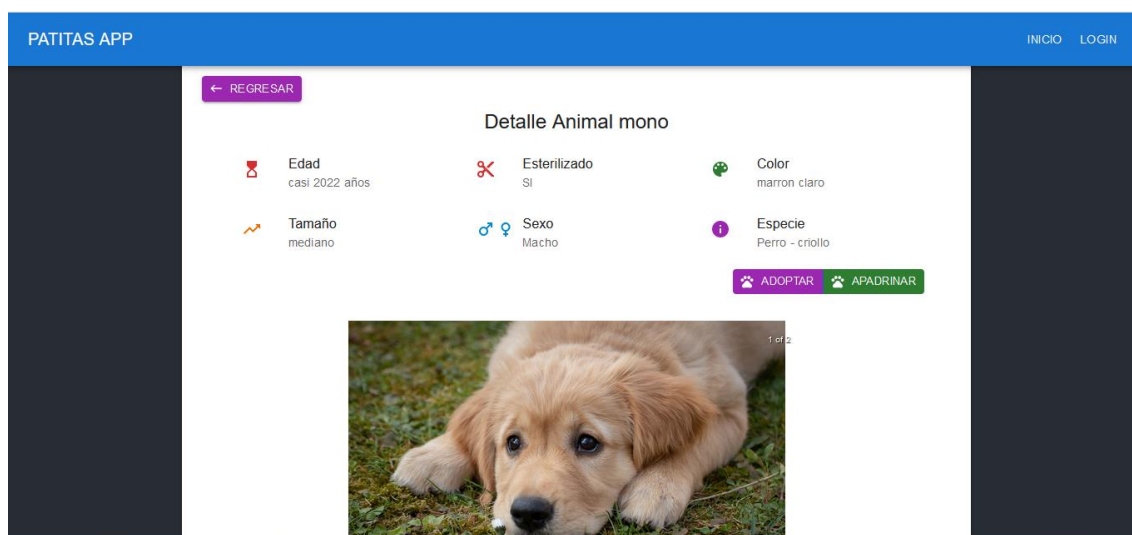


Ilustración 58-Detalle Animal

Fuente: Elaboración propia

Solicitud de adopción

En esta pantalla los usuarios que tiene el interés de adoptar algunos de los animales de la fundación. Deberá proporcionar información personal para que de esta

forma se pongan en contacto para el proceso de adopción (ver Ilustración 59-Solicitud de adopción).

PATITAS APP INICIO LOGIN

← REGRESAR

Formulario de solicitud adopción

Nombres *

Apellidos *

Tipo de documento *

Número de Documento *

Ciudad *

Barrio *

Dirección *

Teléfono *

Correo *

ENVIAR SOLICITUD

Ilustración 59-Solicitud de adopción

Fuente: Elaboración propia

Apadrinar animal

En esta pantalla los usuarios que tenga interés en apadrinar algún animal podrán registrar sus datos (nombres, apellidos, teléfono, correo) con esta información se le enviara información de los pagos quede desee realizar hasta que la mascota se adoptada o el usuario decida cancelar el apadrinamiento (ver Ilustración 60-Formulario de apadrinamiento).

PATITAS APP INICIO LOGIN

← REGRESAR

Formulario de apadrinamiento

Nombres *

Apellidos *

Teléfono *

Correo *

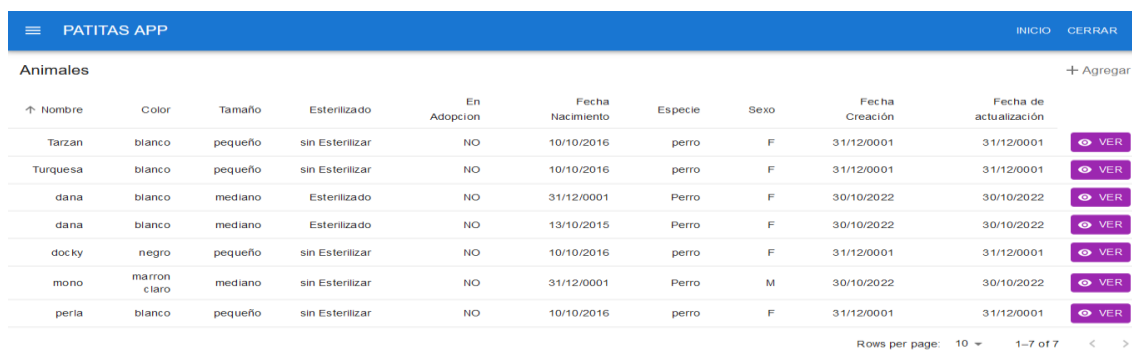
ENVIAR SOLICITUD

Ilustración 60-Formulario de apadrinamiento

Fuente: Elaboración propia

Panel administrativo listar animales

Es esta pantalla solo los usuarios que están autenticados pueden ingresar y listar los animales pertenecientes a la fundación y observar sus datos (ver Ilustración 61- Animales de la fundación).



↑ Nombre	Color	Tamaño	Esterilizado	En Adopción	Fecha Nacimiento	Especie	Sexo	Fecha Creación	Fecha de actualización	
Tarzan	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER
Turquesa	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER
dana	blanco	mediano	Esterilizado	NO	31/12/0001	Perro	F	30/10/2022	30/10/2022	VER
dana	blanco	mediano	Esterilizado	NO	13/10/2015	Perro	F	30/10/2022	30/10/2022	VER
docky	negro	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER
mono	marron claro	mediano	sin Esterilizar	NO	31/12/0001	Perro	M	30/10/2022	30/10/2022	VER
peria	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER

Ilustración 61- Animales de la fundación

Fuente: Elaboración propia

Panel administrativo detalle animal

En esta pantalla solo los usuarios que están autenticados pueden ingresar luego de haber seleccionado en la lista de animales (ver Ilustración 61- Animales de la fundación) se puede apreciar información más detallada del animal

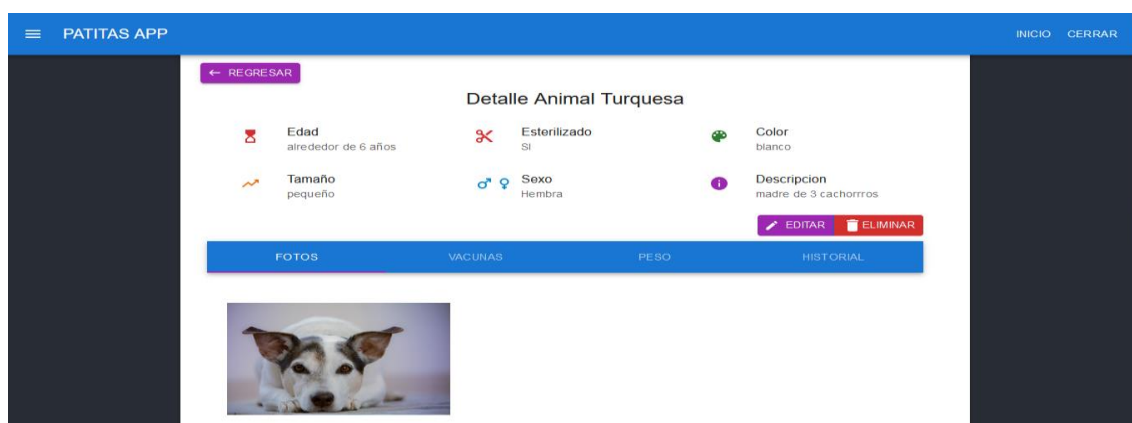


Ilustración 62- Detalle animal

Fuente: Elaboración propia

En esta misma pantalla hay un componente que permite navegar entre otras vistas relacionadas con el animal como sus vacunas (Ver Ilustración 63-Vacunas del Animal) y el control del peso (ver Ilustración 64 - Control de peso del animal)

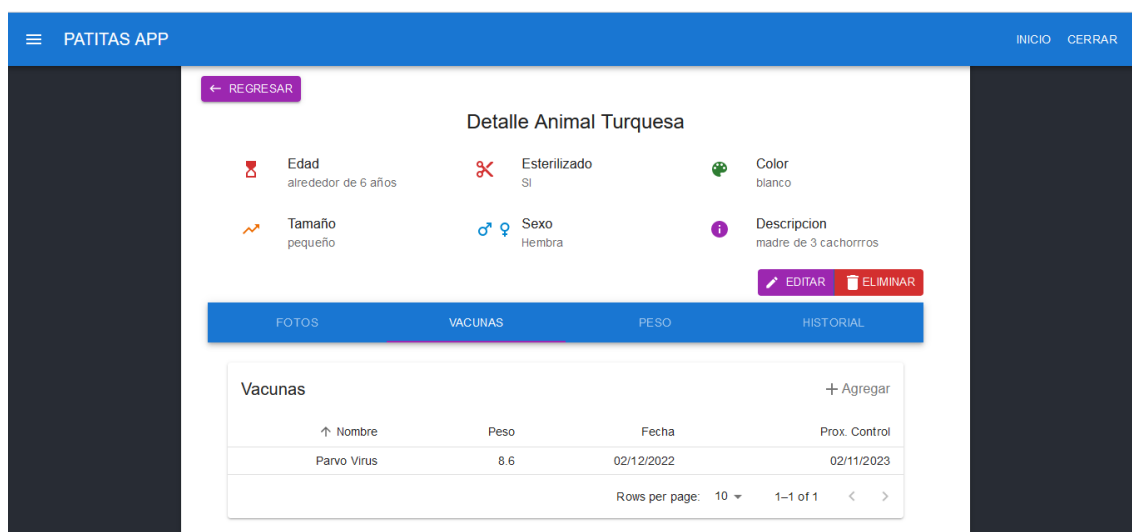


Ilustración 63-Vacunas del Animal

Fuente: Elaboración propia

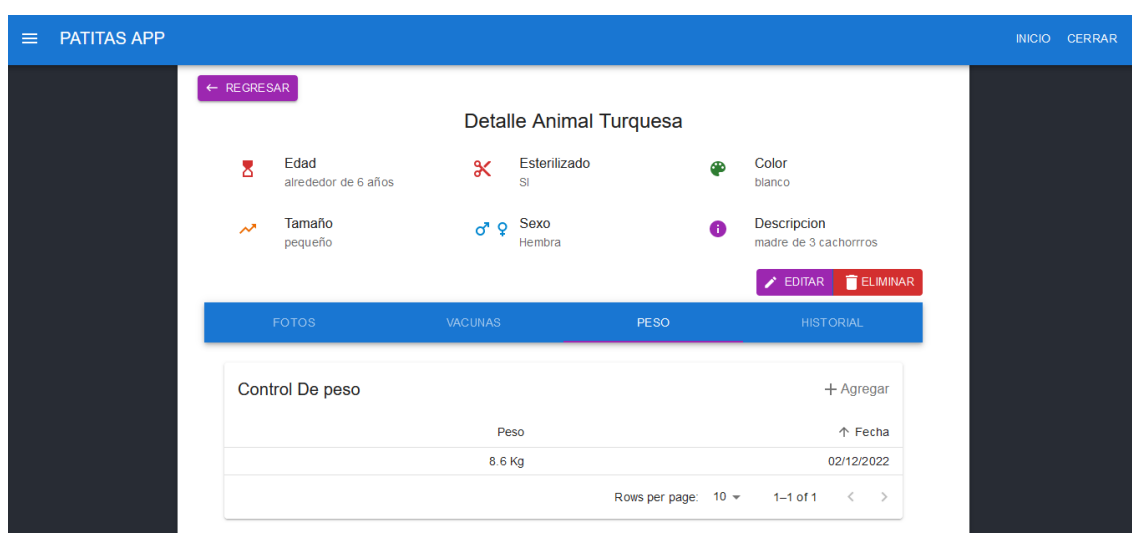


Ilustración 64 - Control de peso del animal

Fuente: Elaboración propia

Agregar animal

En esta pantalla los usuarios autenticados pueden agregar nuevos animales con sus diferentes atributos y de esta forma continuar con los procesos correspondientes (ver Ilustración 65- Agregar animal).

PATITAS APP INICIO CERRAR

← REGRESAR

Formulario para el registro de Animales

Nombre de Animal * Color de Animal *

Tamaño de Animal * Fecha de Nacimiento 02/12/2022

Especie del Animal * Raza del Animal *

Sexo
☐ Hembra ☐ Macho ☐ Esterilizado

ENVIAR

Ilustración 65- Agregar animal

Fuente: Elaboración propia

Eliminar animal

En esta pantalla los usuarios autenticados pueden eliminar animales, confirmando la ventana de eliminación se procede a hacer la petición al back-end desde este componente (ver Ilustración 66- Eliminar animal)

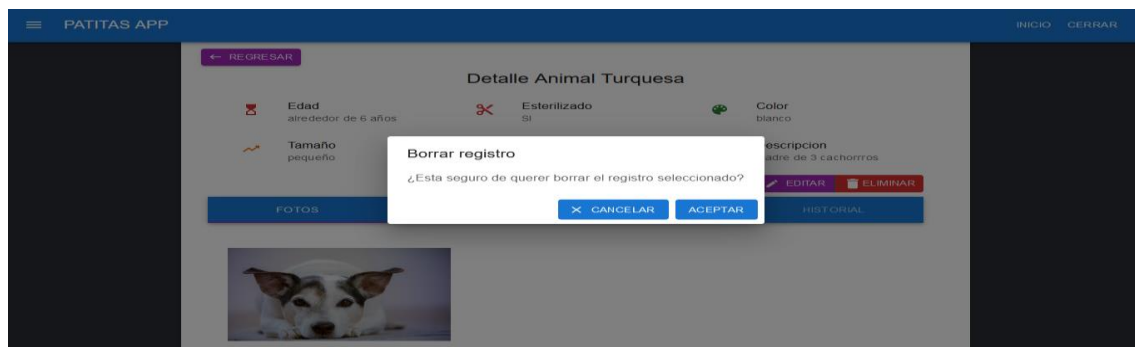


Ilustración 66- Eliminar animal

Fuente: Elaboración propia

Listar solicitudes de adopción

En esta pantalla los usuarios autenticados pueden listar los animales que tienen solicitudes de adopción y al seleccionar y dar clic en ver a más detalle la información del interesado en adoptar al animal (ver Ilustración 67- Solicitudes de adopción).

PATITAS APP								INICIO	CERRAR
Solicitudes de adopción									
↑ Nombre	Color	Tamaño	Esterilizado	En Adopcion	Fecha Nacimiento	Especie	Sexo		
mono	marron claro	mediano	sin Esterilizar	SI	31/12/0001	Perro	M		VER
peria	blanco	pequeño	sin Esterilizar	SI	10/10/2016	perro	F		VER
Rows per page: 10 ▾ 1-2 of 2 < >									

Ilustración 67- Solicitudes de adopción

Fuente: Elaboración propia

Detalle de la solicitud de adopción

En esta pantalla los usuarios autenticados pueden observar los detalles de la solicitud de adopción como es la información del interesado en adoptar (nombre, teléfono, correo, ...) de esta forma la fundación puede ponerse en contacto con él y entrevistarlo para calificarlo si es candidato para adoptar un animal (ver Ilustración 68- Detalle de la solicitud de adopción).

PATITAS APP

INICIO CERRAR

← REGRESAR

Animal mono

Edad

casi 2022 años

Esterilizado

SI

Color

marron claro

Tamaño

mediano



Sexo

Macho

Descripcion

ELIMINAR SOLICITUDES

Solicitudes de adoptantes

↑ Nombres	Apellidos	Documento	Dirección	Teléfono	Email	Estado		
camila	contreras	CC-565656	cuc uta- ciudad jardin-56 vale ve	565656	blade_liger@hotmail.com	inicial	31/12/1969	Cambiar Estad
enyerson	camero	CC-1092343	cuc uta- bocono-av siempre viva	565656565	blade_liger@hotmail.com		31/12/1969	Cambiar Estad
enerson	camero	CC-1092343	cuc uta- bocono-av siempre viva	565656565	blade_liger@hotmail.com		31/12/1969	Cambiar Estad

Ilustración 68- Detalle de la solicitud de adopción

Fuente: Elaboración propia

En esta misma pantalla luego de analizar la información suministrada por el candidato a adoptar puede cambiar el estado de este según los criterios de la fundación (ver Ilustración 69- Cambiar estado de la solicitud de adopción).

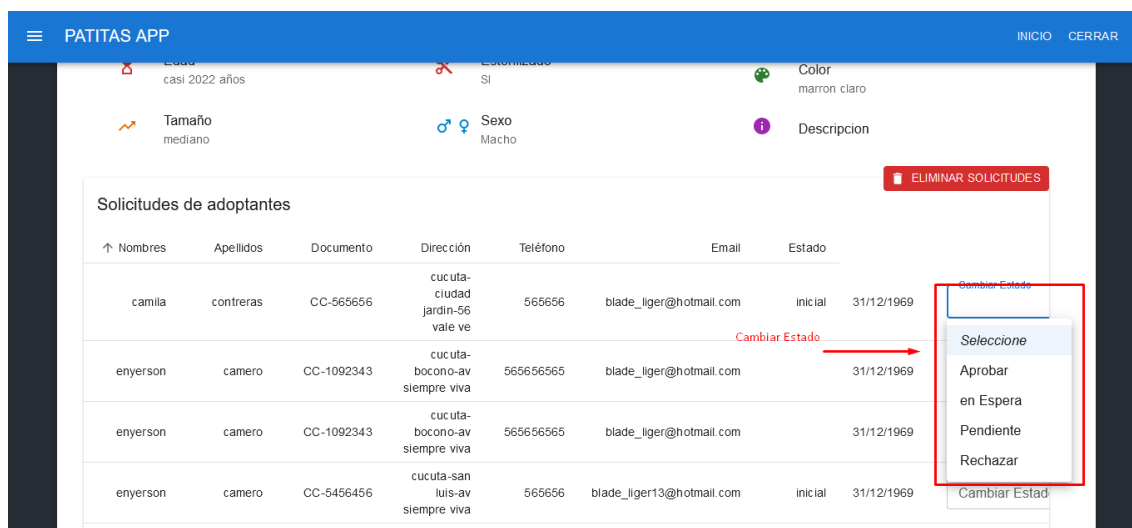


Ilustración 69- Cambiar estado de la solicitud de adopción

Fuente: Elaboración propia

Donación en línea

Esta opción se encuentra en la página de inicio que tiene acceso cualquier persona, la tercera imagen (ver Ilustración 70-Opciones para donar) lleva a los usuarios a un formulario donde pueden ingresar la cantidad con la que desean contribuir (ver Ilustración 71- Formulario de donación).



Ilustración 70-Opciones para donar

Fuente: Elaboración propia

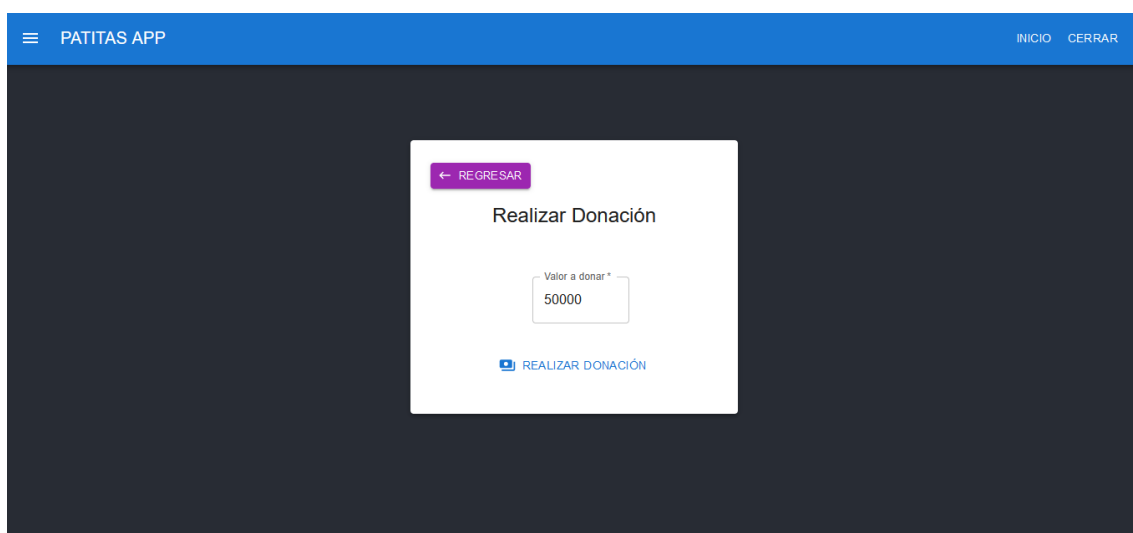


Ilustración 71- Formulario de donación

Fuente: Elaboración propia

Luego de ingresar la cantidad que se desea donar y oprimir el botón de “Realizar Donación” la aplicación llevará al usuario a la pasarela de mercado pago donde podrá recibir la donación (ver Ilustración 72- Opciones de pago mercado pago) estas pantallas con marca de agua son debido que se está realizando el pago en un ambiente de pruebas.

Como ejemplo se escoge el utilizando tarjeta de crédito y van mostrando uno a uno los pasos: seleccionar medio de pago, pago con tarjeta de crédito, información del

titular de la tarjeta de crédito, seleccionar número de cuotas, confirmar pago y pago aprobado (ver 71 a la 76).

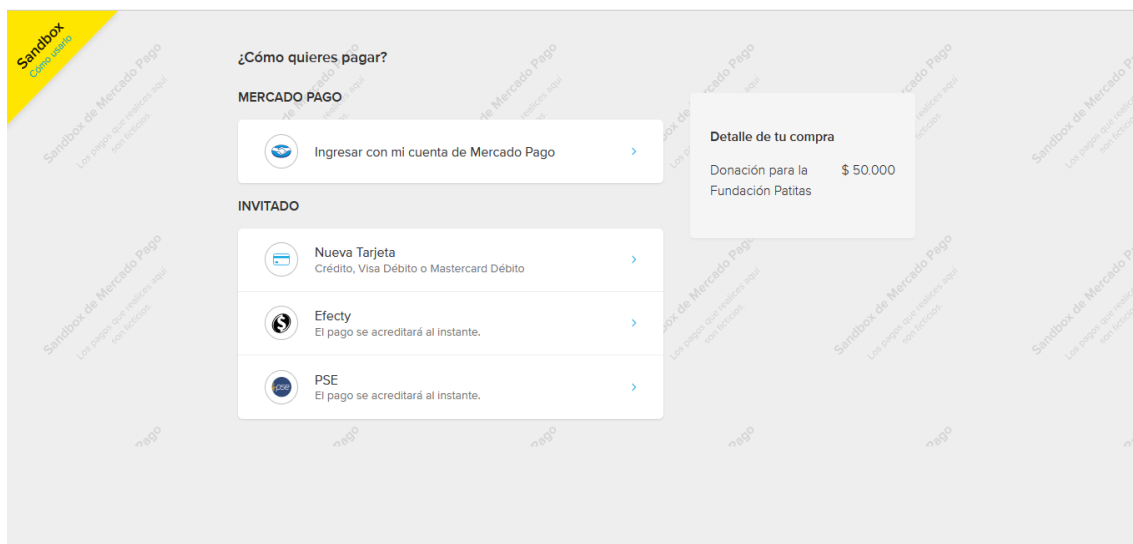


Ilustración 72- Opciones de pago mercado pago

Fuente: (Mercado Pago, 2022)

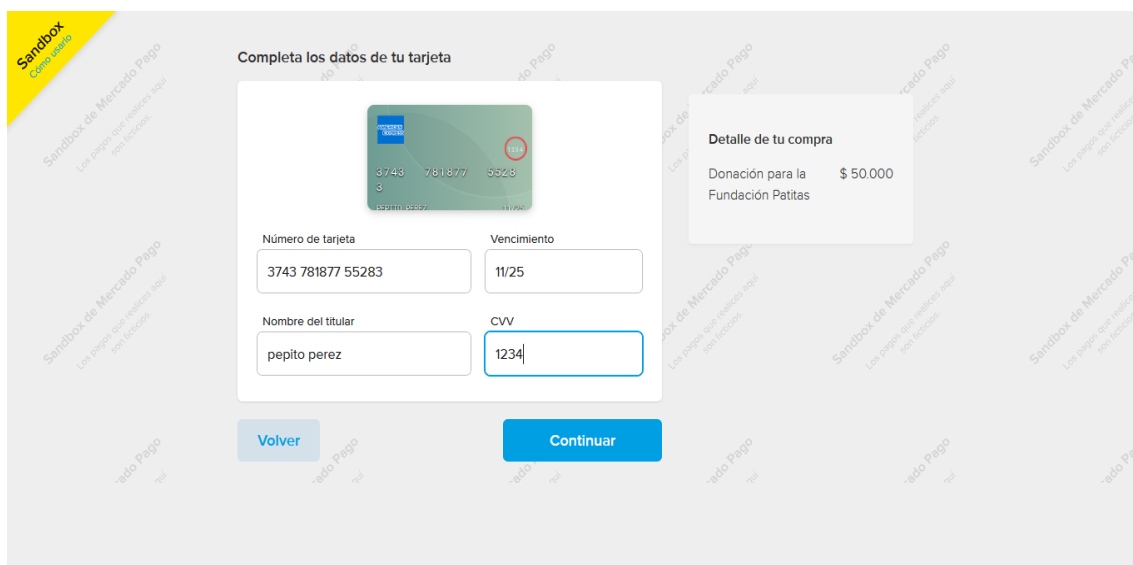


Ilustración 73 - Pago con tarjeta de crédito

Fuente: (Mercado Pago, 2022)

Ahora ingresa el documento del titular

Detalle de tu compra
Donación para la Fundación Patitas \$ 50.000

Tipo: C.C. # 123456

[Volver](#) [Continuar](#)

Ilustración 74- Dato del titular de la tarjeta de crédito

Fuente: (Mercado Pago, 2022)

Elige las cuotas que quieras

1x	\$ 50.000	
2x	\$ 25.000	Tu banco aplicará los intereses
3x	\$ 16.667	Tu banco aplicará los intereses
4x	\$ 12.500	Tu banco aplicará los intereses
5x	\$ 10.000	Tu banco aplicará los intereses
6x	\$ 8.333	Tu banco aplicará los intereses
12x	\$ 4.167	Tu banco aplicará los intereses

Detalle de tu compra
Donación para la Fundación Patitas \$ 50.000

Ilustración 75- Selección de numero de cuotas

Fuente: (Mercado Pago, 2022)

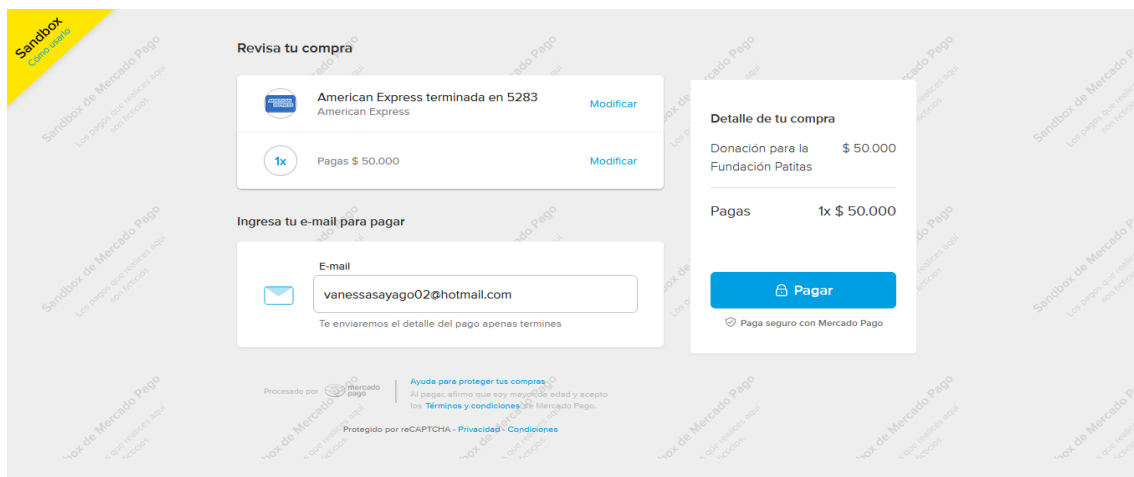


Ilustración 76- Confirmar pago

Fuente: (Mercado Pago, 2022)

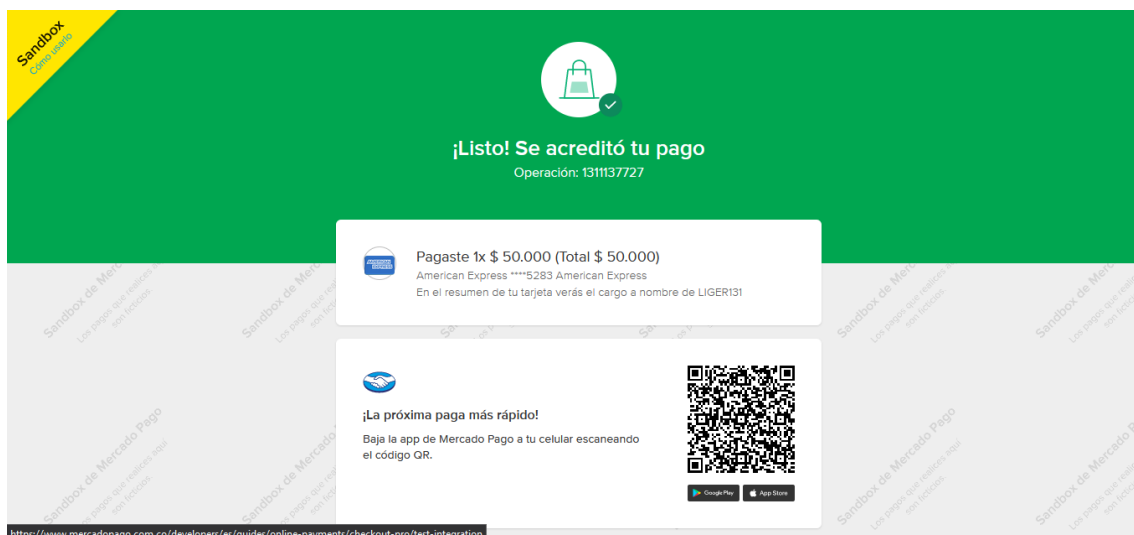


Ilustración 77- Transacción realizada con éxito en mercado pago

Fuente: (Mercado Pago, 2022)

Modulo inteligente

Para poder hacer uso del algoritmo KNN se realiza por medio de la librería **golearn** (ver Ilustración 78 - importar librerías **golearn**) que realiza su implementación de forma sencilla y para quienes tiene poca experiencia en el uso de estas herramientas.

```

adopt.services.go X
services > adopt.services.go
1 package services
2
3 import (
4     "context"
5     "fmt"
6     "log"
7     "os"
8     "time"
9
10    "github.com/edcamero/api-go/models"
11    "github.com/edcamero/api-go/util"
12    "github.com/go-gota/gota/dataframe"
13    "github.com/go-gota/gota/series"
14    "github.com/sjwhitworth/golearn/base"
15    "github.com/sjwhitworth/golearn/knn"
16    "go.mongodb.org/mongo-driver/bson"
17    "go.mongodb.org/mongo-driver/bson/primitive"
18    "go.mongodb.org/mongo-driver/mongo"
19    "go.mongodb.org/mongo-driver/mongo/options"
20 )
21

```

Importar librerías de golearn

Ilustración 78 - importar librerías golearn

Fuente: Elaboración propia

Una vez estamos importando las librerías se crea el método que leer la información (ver Ilustración 79 - información de aprendizaje) que sirve para el aprendizaje del algoritmo que es llamado en la función GetClueIA. Luego de leer la información esta es pasada por el algoritmo KNN que crea nueva información a partir de la lectura.

A	B	C	D	E	F	G	H	I	J	K
sexo	raza	edad	color	tamaño	especie	sexo-adop	tamaño-vivi	estado-civil	edad-adop	estrato
Hembra	Poodle		1 Caoba	grande	Canino	M	finca	soltero	35	4
Hembra	Boston terrie		3 Caoba	mediano	Canino	M	casa	soltero	57	4
Macho	Mestizo		2 Caoba	grande	Canino	F	finca	casado	22	2
Macho	Shih Tzu		3 Negro	pequeño	Canino	F	apartamento	casado	57	2
Hembra	Shih Tzu		4 Negro	mediano	Canino	M	casa	divorciado	41	3
Macho	Shih Tzu		1 Caoba	pequeño	Canino	F	apartamento	divorciado	28	2
Hembra	Bull Dog Ingl		1 Blanco	grande	Canino	F	finca	divorciado	56	4
Macho	Bull Dog Ingl		1 Blanco	mediano	Canino	M	casa	casado	34	2
Hembra	Poodle		3 Negro	mediano	Canino	F	casa	soltero	24	5
Macho	Poodle		1 Caoba	mediano	Canino	F	casa	divorciado	38	2
Macho	Mestizo		1 Caoba	grande	Canino	M	finca	divorciado	19	1
Macho	Pincher		1 Marrón	grande	Canino	M	finca	soltero	52	3
Macho	Pincher		3 Caoba	grande	Canino	F	finca	casado	33	4
Macho	Cocker		2 Caoba	mediano	Canino	M	casa	divorciado	19	2
Hembra	Poodle		4 Marrón	grande	Canino	M	finca	soltero	48	2
Hembra	Shih Tzu		2 Caoba	mediano	Canino	F	casa	divorciado	28	2

Ilustración 79 - información de aprendizaje

```

func GetClueIA(adopt *models.AdoptanteClue) models.AnimalResponseIA {
    directory := "./data-mascotas2.csv"
    filePathResponse := "./respuesta.csv"

    if err := os.Truncate(filePathResponse, 0); err != nil {
        log.Printf("Failed to truncate: %v", err)
    }
    rawData, err := base.ParseCSVToInstances(directory, true)
    if err != nil {
        panic(err)
    }

    //Initialises a new KNN classifier
    cls := knn.NewKnnClassifier("euclidean", "linear", 2)

    //Do a training-test split
    trainData, testData := base.InstancesTrainTestSplit(rawData, 0.50)
    cls.Fit(trainData)

    base.SerializeInstancesToCSV(testData, filePathResponse)
}

```

Parametros ingresados por el solicitante

Ruta de los archivos

Lectura del archivo para el aprendizaje

Definición del algoritmo de aprendizaje

Genera nuevos datos apartir del aprendizaje

Almacena los nuevos datos

Ilustración 80- Función de búsqueda parte 1

Fuente: Elaboración propia

Luego de generar la información es almacenada para su posterior lectura para poder aplicarle filtro (ver Ilustración 81- Función de búsqueda parte 2) con la información enviada por el usuario ya que la estructura de dato que maneja no permite la aplicación de filtros.

```

result, err := os.Open(filePathResponse) ← Leer los datos generados
df := dataframe.ReadCSV(result)
df = df.Filter( ← Filtrar datos
    dataframe.F{Colname: "tamaño-vivienda", Comparator: series.Eq, Comparando: adopt.Vivienda},
    dataframe.F{Colname: "sexo-adoptante", Comparator: series.Eq, Comparando: adopt.Sexo},
    dataframe.F{Colname: "estado-civil", Comparator: series.Eq, Comparando: adopt.EstadoCivil},
    dataframe.F{Colname: "edad-adoptante", Comparator: series.Greater, Comparando: adopt.Edad + 5},
    dataframe.F{Colname: "edad-adoptante", Comparator: series.LessEq, Comparando: adopt.Edad - 5},
    dataframe.F{Colname: "estrato", Comparator: series.LessEq, Comparando: adopt.Estrato},
)
fmt.Println(testData)

var results models.AnimalResponseIA

results.Color = df.Col("color").Elem(0).String()
results.Tamaño = df.Col("tamaño").Elem(0).String() ← Obtener recomendación
results.Especie = df.Col("especie").Elem(0).String()
results.Raza = df.Col("raza").Elem(0).String()

return results
}

```

Ilustración 81- Función de búsqueda parte 2

Fuente: Elaboración propia

La función retorna atributos de los animales cuyos datos ingresados por el usuario concuerda con los datos generados con por el algoritmo KNN estos datos son retornados y usados por la función GetClue para realizar la consulta en los animales disponibles en la fundación obteniendo los datos de un animal y retornándolos al usuario. (ver Ilustración 83 - Controlador búsqueda sugerida)

```

func (service petsService) GetClue(adopt *models.AdoptanteClue) (models.AnimalDetail, error) {
    var petValue models.AnimalDetail

    filterObject := GetClueIA(adopt) ← Llamado de función inteligente

    fmt.Println(filterObject)

    filter := bson.D{ ← Define filtro para mongoDB
        {Key: "tamaño", Value: filterObject.Tamaño},
        {Key: "especie", Value: filterObject.Especie},
        {Key: "estado", Value: true},
        {Key: "en_adopcion", Value: true},
    }

    err := service.animalCollection.FindOne(nil, filter).Decode(&petValue) ← Realiza búsqueda basado en los datos obtenidos
    if err == mongo.ErrNoDocuments {
        return petValue, err
    }
    return petValue, err
}

```

Ilustración 82 - servicio búsqueda sugerida

Fuente: Elaboración propia

```

func (handler *PetsService) GetClue(ctx iris.Context) {

    dataAdopt := new(models.AdoptanteClue)
    err := ctx.ReadJSON(dataAdopt)

    if err != nil {
        util.FailJSON(ctx, iris.StatusBadRequest, err, "Malformed request payload")
        return
    }

    pet, err := handler.service.GetClue(dataAdopt)
    if err != nil {
        fmt.Println(err)
        ctx.StopWithStatus(iris.StatusNotFound)
        return
    }

    ctx.JSON(pet)
}

```

Obteniendo la información de la solicitud

Llamando a la función de sugerencia de mascota

Ilustración 83 - Controlador búsqueda sugerida

Fuente: Elaboración propia

Se agrega el botón de sugerir mascota en la página principal (ver Ilustración 84 - Ingresar sugerir mascota) donde cualquier usuario puede oprimir, y ingresar a una pantalla donde se le solicitarán datos como estado civil, edad, tipo de vivienda, sexo y estrato (ver Ilustración 85 - Formulario para sugerencia de mascota) que serán usados por el algoritmo para la obtener la mejor recomendación posible.

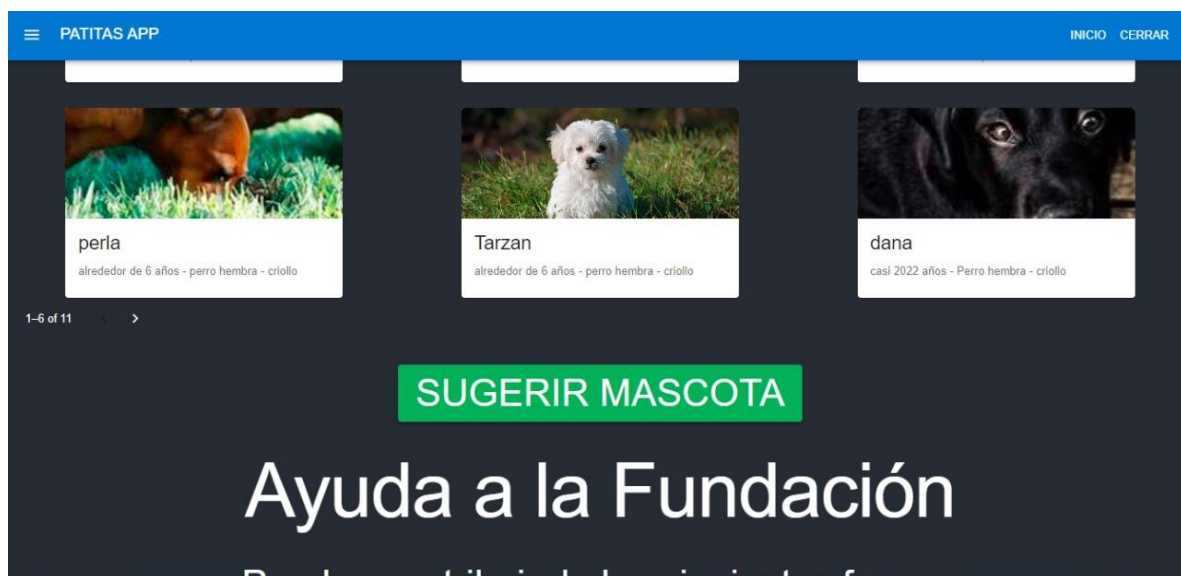


Ilustración 84 - Ingresar sugerir mascota

 The screenshot shows the 'Formulario de Sugerencia de Mascota' in the PATITAS APP. The form is displayed on a dark background. At the top of the form, there's a purple button labeled '← REGRESAR'. Below it, the title 'Formulario de Sugerencia de Mascota' is centered. The form contains several input fields: 'Vivienda *' with a dropdown menu showing 'apartamento'; 'Sexo *' with a dropdown menu showing 'Masculino'; 'Estado Civil *' with a dropdown menu showing 'casado'; 'Estrato *' with a dropdown menu showing '2'; and 'Edad *' with a text input field showing '30'. At the bottom of the form, there's a blue button labeled 'ENVIAR SOLICITUD'.

Ilustración 85 - Formulario para sugerencia de mascota

La información retornada por el controlador es de un animal que se visualiza en la misma pantalla de detalle animal público. Ya el usuario tomara a decisión de adoptar.

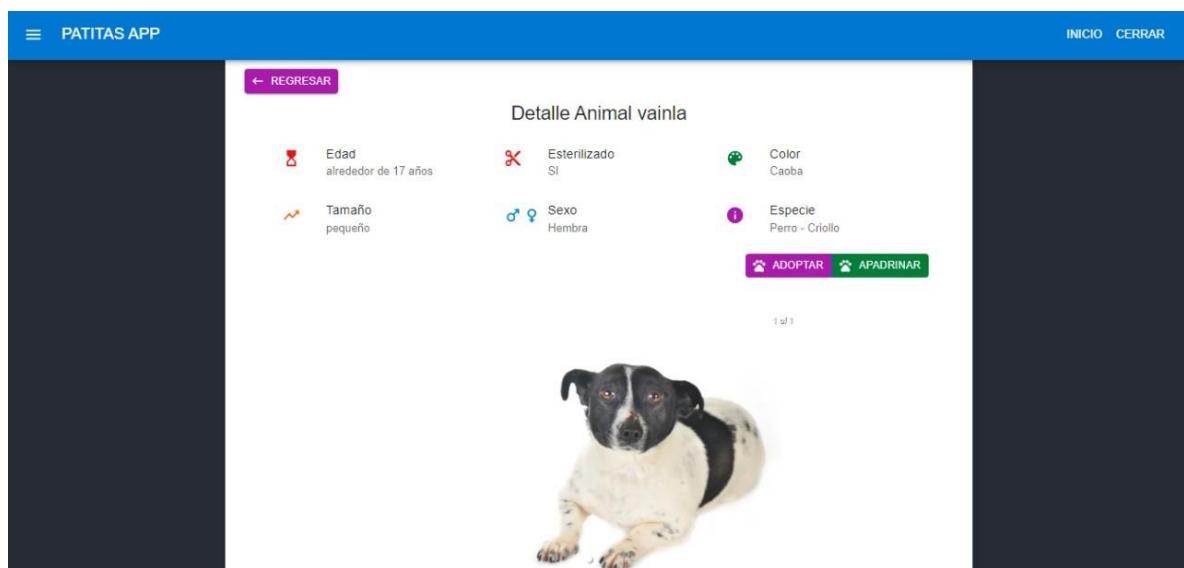


Ilustración 86 - animal sugerido

Desarrollo Back-end

Para desarrollar el aplicativo en el servidor o back-end, contamos con todo lo que no es visible para el usuario, pero es importante para el funcionamiento del sistema, ya que permite el acceso a la información solicitada y la correcta entrada de datos. Esta cubre la lógica, los métodos, las funciones, el uso de la biblioteca, el manejo de datos, el enrutamiento, la seguridad, los módulos, los controladores y el middleware, y mucho más.

A continuación, se hará una explicación de algunos componentes en el proceso del desarrollo de la aplicación.

Controllers o controladores

Los controllers nos permite agrupar la lógica, además comunicar las rutas con los modelos, demás clases y servicios para dar una respuesta al usuario, estos se almacenan en el directorio o carpeta controllers, hay que tener en cuenta que para crear un controlador se hace de forma manual ya que aunque iris cuenta con un generador de código este no está disponible para Windows, en los controladores se crean las funciones que son los puntos de entrada para realizar las operaciones típicas de creación, lectura, actualización y eliminación ("CRUD").

```

3 import (
4     "log"
5     "path/filepath"
6
7     "github.com/edcamero/api-go/models"
8     "github.com/edcamero/api-go/services"
9     "github.com/edcamero/api-go/util"
10    "github.com/kataras/iris/v12"
11 )
12
13 type PetsService struct {
14     service      services.PetsService
15     serviceUpload services.UploadService
16 }
17
18 func NewPetsController(service services.PetsService, serviceUpload services.UploadService) *PetsService {
19     return &PetsService{service: service, serviceUpload: serviceUpload}
20 }
21
22 func (handler *PetsService) Count(ctx iris.Context) {
23     count, err := handler.service.Count(nil)
24     if err != nil {
25         ctx.StopWithStatus(iris.StatusInternalServerError)
26         return
27     }
28     ctx.JSON(count)
29 }
30
31 func (handler *PetsService) GetAll(ctx iris.Context) {
32     page, err := ctx.Params().GetInt64("page")
33     base, err := ctx.Params().GetInt64("base")
34     if err != nil {
35         log.Print(err)
36         ctx.StopWithStatus(iris.StatusBadRequest)
37         return
38     }
39 }

```

Importación de librerías

Definición de controlador

Intancia el controlador y inyecta los servicios a usar

Función del controlador

Funcion para obtener los parametros enviados al controlador

Ilustración 87- Controlador en Iris

Fuente: Elaboración propia

En el segmento de código anterior (Ilustración 87- Controlador en Iris) se muestra un controlador en donde se almacenan los métodos y la lógica para ser llamados al momento de hacer las peticiones HTTP, en la parte superior se observa la palabra `import` que hace referencia a los llamados de algunas librerías y modelos que son usados en las funciones de este controlador.

Models o modelos

Los Modelos son componentes importantes que facilitan la interacción con nuestra base de datos para realizar cualquier tipo de consulta ; ya sea insertar, obtener, modificar y borrar, estos se almacenan en el directorio o carpeta `app/Models`; cabe resaltar que para crear un modelo se hace mediante el comando `go generate` estos modelos ofrecen una gran ventaja ya que incluyen etiquetas como `json` que permite al struct presentar un atributo en formato json con diferente nombre o nomenclatura al recibir o responder peticiones HTML , lo mismo sucede con la etiqueta `bson` permite trabajar con campos

en la base de datos con diferentes nomenclatura como se puede ver en Ilustración 88- Modelo en Golang donde el atributo Nombre está en mayúscula su primera letra (esto lo identifica como un atributo público en Golang) pero no se desea que al responder peticiones http este se envíe de esta forma es por eso que se agrega la etiqueta `json:"nombre"` y en la base de datos mongoBD el campo esta como nombre. pero para poder trabajar con el es necesario la etiqueta `bson:"nombre"`

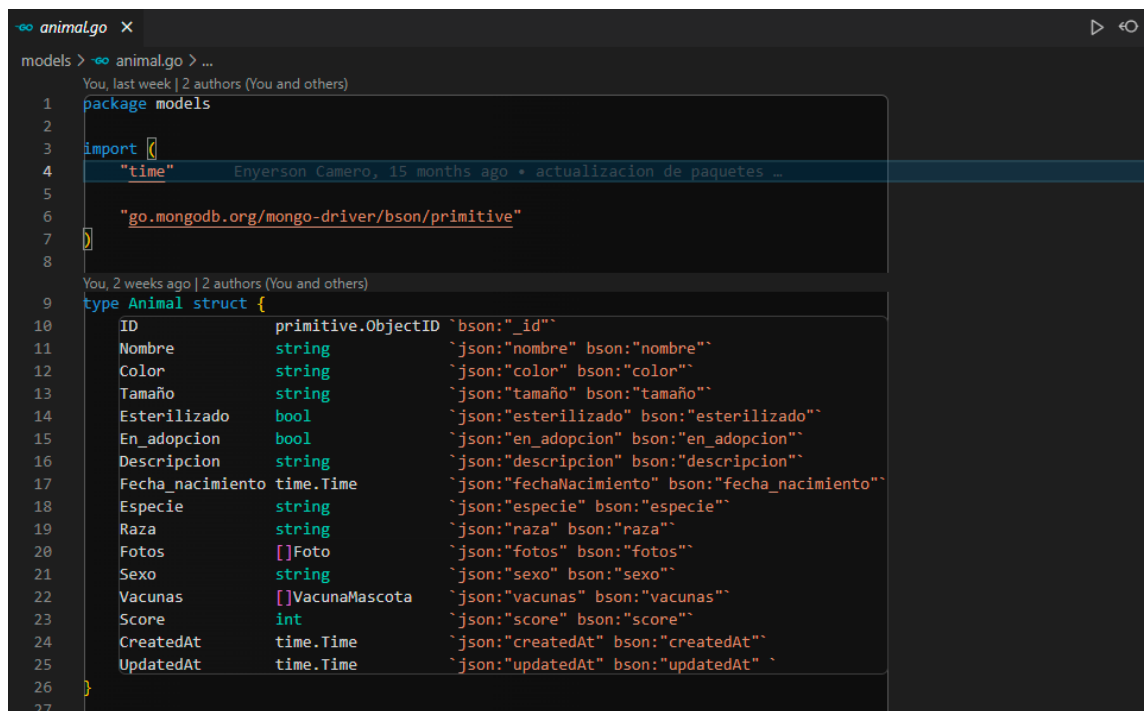


Ilustración 88-Modelo en Golang

Fuente: Elaboración propia

Services o servicios

Los services es donde se encuentra la mayor parte de la lógica que realiza la conexiones con la base de datos y de esta forma poder ser llamados desde los controladores. A los servicios se le define dentro de una interface para que sus métodos no estén fuera de esa definición como se puede apreciar en la Ilustración 89 - Servicio en Golang.

Cada servicio es solo responsable de una colección de la base de datos que es inyectada en el momento de crear el servicio como se puede apreciar en la Ilustración 90- Características del servicio.

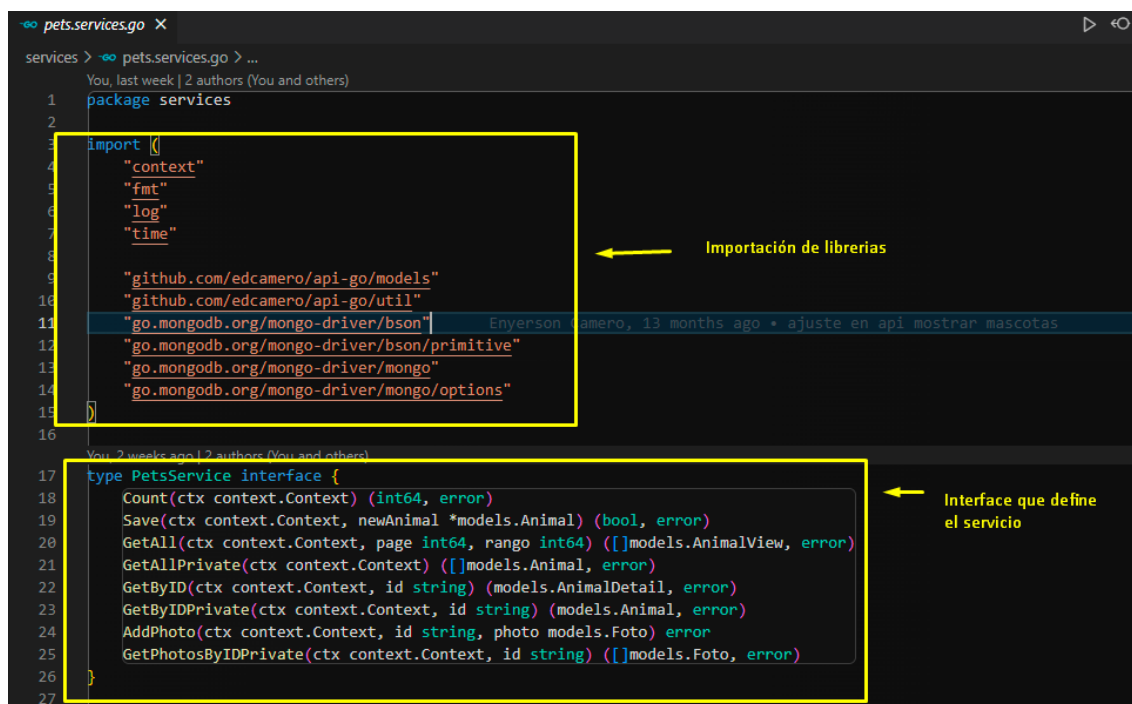


Ilustración 89 - Servicio en Golang

Fuente: Elaboración propia

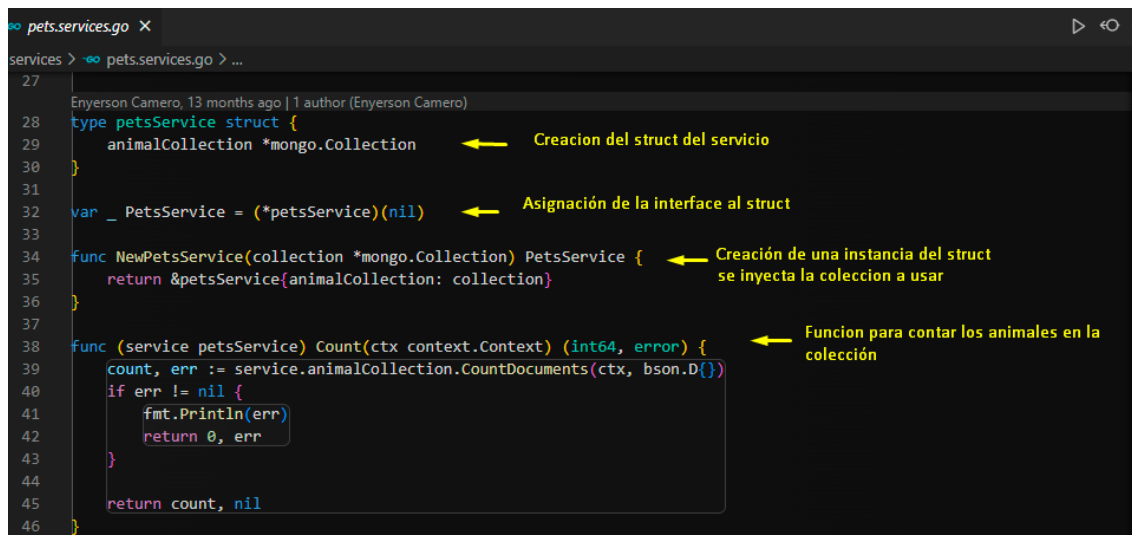


Ilustración 90- Características del servicio

Fuente: Elaboración propia

Las consultas que se tiene que realizar en las colecciones muchas veces no son tan sencillas como la vista en la función `Count` de la Ilustración 90- Características del

servicio. En muchas ocasiones son bastantes complejas y tediosas como se puede apreciar en la Ilustración 91- Consulta MongoDB desde Golang. Donde se debe crear una proyección diciéndole que campo deben ser devueltos y luego pasar esta proyección como parámetro a la función que proporciona la librería de mongo-drive

```

47
48 func (service petsService) GetAll(ctx context.Context, page int64, rango int64) ([]models.AnimalView, error) {
49
50     var endItem int64 = page * rango
51     var startItem int64 = 0
52     if page > 0 {
53         startItem = (page - 1) * rango
54     }
55
56     projection := bson.D{
57         primitive.E{Key: "nombre", Value: 1},
58         primitive.E{Key: "color", Value: 1},
59         primitive.E{Key: "tamaño", Value: 1},
60         primitive.E{Key: "esterilizado", Value: 1},
61         primitive.E{Key: "descripcion", Value: 1},
62         primitive.E{Key: "fecha_nacimiento", Value: 1},
63         primitive.E{Key: "especie", Value: 1},
64         primitive.E{Key: "raza", Value: 1},
65         primitive.E{Key: "fotos", Value: bson.D{primitive.E{Key: "$slice", Value: 1}}},
66         primitive.E{Key: "sexo", Value: 1},
67     }
68
69     findOptions := options.Find().SetProjection(projection).SetSkip(startItem).SetLimit(endItem).SetSort(bson.D{
70         cursor, err := service.animalCollection.Find(ctx, bson.D{}, findOptions)
71     if err != nil {
72         fmt.Println(err)
73         return nil, err
74     }
75     defer cursor.Close(ctx)
76
77     var results []models.AnimalView
78
79     for cursor.Next(ctx) {
80         if err = cursor.Err(); err != nil {
81             fmt.Println("este error cursor")
82             return nil, err
83         }
84     }
85 }

```

Establece los valores para la paginación

Define que campos serán mostrados en la consulta

Ejecución de la consulta

Si la consulta resulta en un error regresa el error que será tratado en el controlador

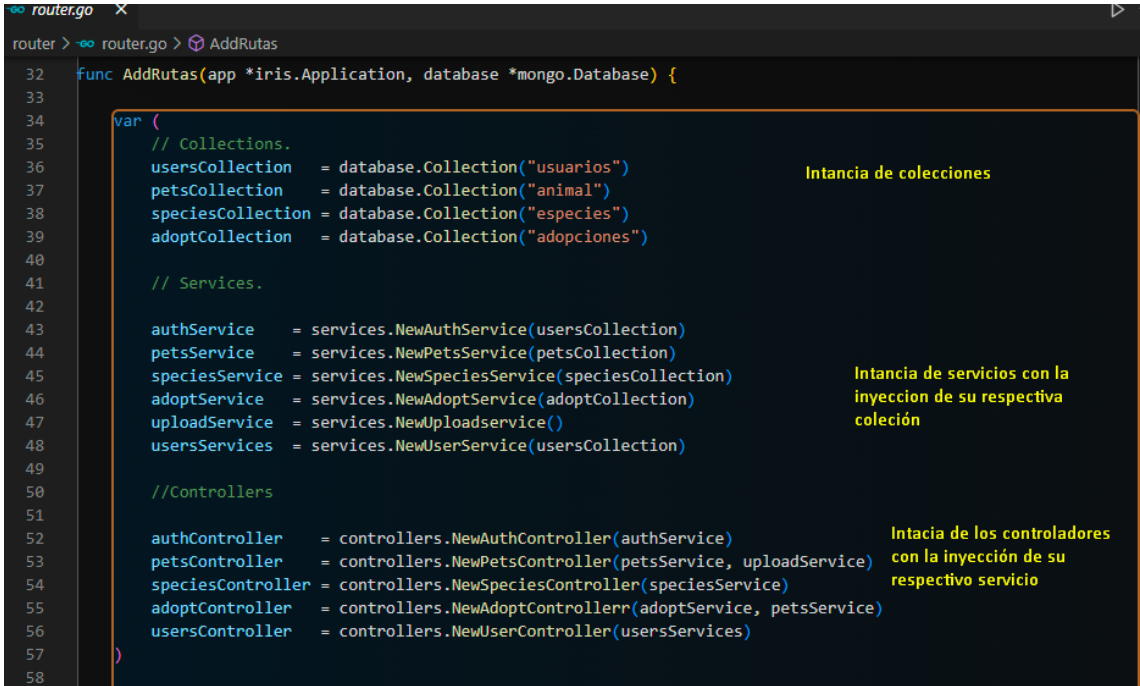
Recorre el resultado para crear el arreglo que será regresado como respuesta

Ilustración 91- Consulta MongoDB desde Golang

Fuente: Elaboración propia

Routes o rutas

Las rutas son responsables de atender el flujo de solicitudes HTTP hacia y desde el usuario. Se guardan en la raíz del proyecto en una carpeta llamada "routes/". En el archivo "route.go" definimos las rutas, las cuales pueden ser de tipo Get y Post, recordando que para utilizar el sistema de rutas se debe importar las colecciones, servicios y controladores, este último es utilizado para el enrutamiento (ver Ilustración 92 - Inyección de dependencias E ilustración 83 - Definición de rutas).



```

32 func AddRutas(app *iris.Application, database *mongo.Database) {
33
34     var (
35         // Collections.
36         usersCollection = database.Collection("usuarios")
37         petsCollection  = database.Collection("animal")
38         speciesCollection = database.Collection("especies")
39         adoptCollection  = database.Collection("adopciones")
40
41         // Services.
42
43         authService = services.NewAuthService(usersCollection)
44         petsService  = services.NewPetsService(petsCollection)
45         speciesService = services.NewSpeciesService(speciesCollection)
46         adoptService  = services.NewAdoptService(adoptCollection)
47         uploadService = services.NewUploadService()
48         usersServices = services.NewUserService(usersCollection)
49
50         //Controllers
51
52         authController = controllers.NewAuthController(authService)
53         petsController  = controllers.NewPetsController(petsService, uploadService)
54         speciesController = controllers.NewSpeciesController(speciesService)
55         adoptController  = controllers.NewAdoptController(adoptService, petsService)
56         usersController  = controllers.NewUserController(usersServices)
57     )
58

```

Intancia de colecciones

Intancia de servicios con la inyeccion de su respectiva colección

Intacia de los controladores con la inyección de su respectivo servicio

Ilustración 92 - Inyección de dependencias

Fuente: Elaboración propia

```

69
70 api := app.Party("/api")
71 api.Post("/login", authController.Login)
72 api.Get("/pets/page/{page:uint64}/base/{base:uint64}", petsController.GetAll)
73 api.Get("/pets/count", petsController.Count)
74 api.Get("/pets/{id:string}", petsController.GetByID)
75 api.Post("/pets/{id:string}/adopt", adoptController.Save)
76 api.Use(util.Verify())
77 api.Use(j.Serve)
78 api.Get("/refresh", authController.RefreshToken)
79
80 api.Post("/adoptante/registrar", Controllers.RegisterAdoptante)
81
82 adopApi := api.Party("/adoptante")
83 adminApi := api.Party("/admin", j.Serve, Controllers.AuthenticatedAdmin)
84
85 adopApi.Get("/", Controllers.AuthenticatedAdoptante)
86
87 adminApi.Post("/registrar", Controllers.RegisterAdmin)
88
89 api.Get("/user/{id:uint64}", Controllers.GetUser)
90 api.Get("/user", Controllers.AllUsers)
91 api.Post("/user", Controllers.StoreUsers)
92 api.Put("/user/{id:uint64}", Controllers.UpdateUser)
93 api.Delete("/user/{id:uint64}", Controllers.DeleteUser)
94 // routes pets private
95
96 //pets
97 adminApi.Get("/pets", petsController.GetAllPrivate)
98
99 adminApi.Post("/pets", petsController.SavePrivate)
100 adminApi.Get("/pets/{id:string}", petsController.GetByIDPrivate)
101 adminApi.Get("/pets/{id:string}/photo", petsController.GetPhotosByID)
102 adminApi.Post("/pets/{id:string}/photo/upload", iris.LimitRequestBodySize(10<<20), petsController.UploadFile)
103

```

Ilustración 93 - Definición de rutas

Fuente: Elaboración propia

Desarrollo de los Requerimiento no funcionales

A continuación, se describen las principales características no funcionales que debe contener el sistema de información web para la adopción compatible y el apadrinamiento de animales en condición de calle para la ciudad de Cúcuta

Seguridad

Se requiere de la implementación de políticas de seguridad además se deben considerar los siguientes aspectos:

Identificación y Autentificación: la autentificación es requerida para usuarios de rol fundación que tiene la responsabilidad de lectura, administración y almacenamiento de los datos en el sistema. Para las peticiones que requieren ser realizadas por usuarios autenticados en la aplicación se agrega un token de seguridad en el momento que el

Y para evitar que un token que se ha generado en un ambiente de pruebas se utilizado para realizar peticiones en un ambiente de producción entra en juego la clave secreta (ver Ilustración 96 - Generación del token) o secretkey que es una variable de entorno y que debe ser diferentes en ambos ambientes de esta forma un token del ambiente de pruebas es invalido en un ambiente de producción.

```

3  import (
4      "fmt"
5      "time"
6
7      "github.com/edcamero/api-go/environment"
8      "github.com/edcamero/api-go/models"
9      "github.com/kataras/iris/v12"
10     "github.com/kataras/iris/v12/middleware/jwt"
11 )
12
13 const (
14     accessTokenMaxTime = 1000 * time.Minute ← Tiempo de vida del token
15     refreshTokenMaxAge = 1200 * time.Minute
16 )
17
18 var (
19     secretKey = []byte(environment.SECRETKEY) ← Clave secreta
20     signer    = jwt.NewSigner(jwt.HS256, secretKey, accessTokenMaxTime)
21     verifier  = jwt.NewVerifier(jwt.HS256, secretKey)
22 )
23
24 func GenerateToken(user models.Claims) (jwt.TokenPair, error) { ← Función para generar token
25
26     claims := user
27     refreshClaims := jwt.Claims{Subject: user.Id}
28     tokenPair, err := signer.NewTokenPair(claims, refreshClaims, refreshTokenMaxAge)
29     if err != nil {
30         fmt.Println(err)
31         return tokenPair, err
32     }
33     return tokenPair, nil
34 }

```

Ilustración 96 - Generación del token

Fuente: Elaboración propia

Tiempo de respuesta

El equipo de desarrollo de Iris Web Framework carga regularmente pruebas de rendimiento de sus herramientas, este conjunto de pruebas está diseñado para comparar todo el manejo de solicitudes HTTP en Go Web Frameworks. En el gráfico (Ilustración

97 - Pruebas de velocidad de Iris) puede ver los hechos que muestran el sistema web Iris, que es probablemente el sistema web más rápido para el desarrollo del lenguaje GO.

Benchmarks: Jun 20, 2022 at 8:17pm (UTC)

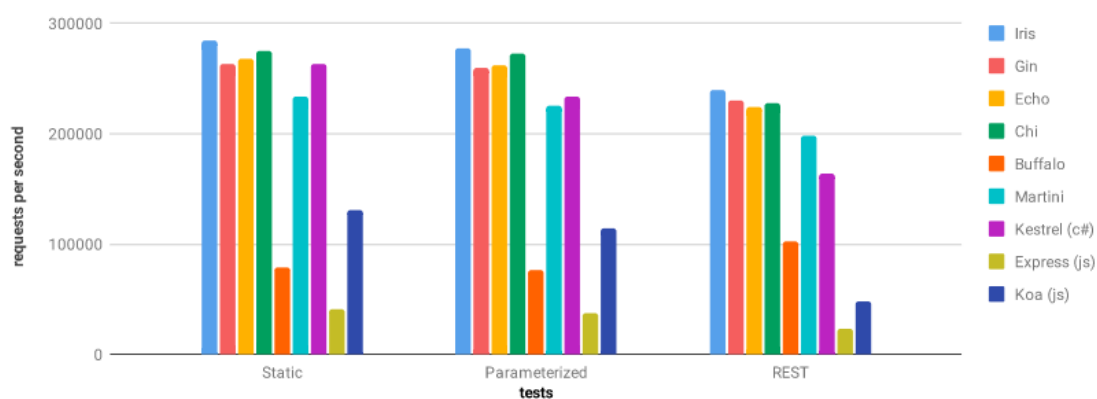


Ilustración 97 - Pruebas de velocidad de Iris

Fuente: (Iris, 2022)

En la aplicación desarrollada al hacer una petición para obtener los animales al siguiente endpoint: <http://localhost:8080/api/admin/pets> solo toma 11 milisegundo (ver Ilustración 98 - Velocidad de una solicitud de la aplicación) esto es debido a la combinación de Iris junto mongoDB

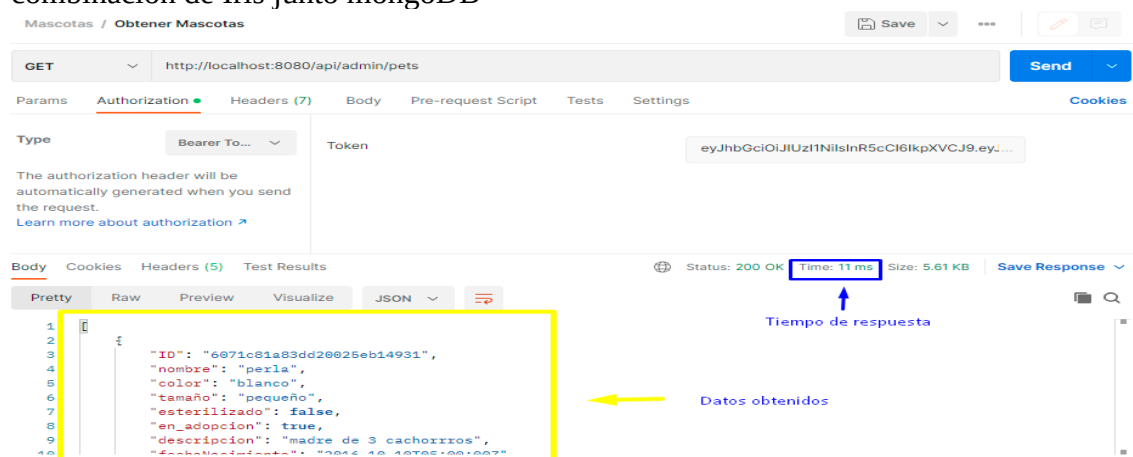


Ilustración 98 - Velocidad de una solicitud de la aplicación

Fuente: Elaboración propia

Amigable con el usuario

El diseño de las interfaces grafica fue realizado utilizando la librería MUI que es muy característica por apegarse a los principios de Merial Design los cuales son unos

principios que intenta de aproximarse a la realidad jugando con las profundidades, los bordes, las sombras y los colores. (Material Design, 2022) y que brinda una buena experiencia de uso al usuario.

Datos personales

Cualquier organización generalmente debe reconocer los derechos de los usuarios de Habeas Data para actualizar, modificar y eliminar la información recopilada sobre ellos en bases de datos y archivos; La empresa se compromete a mantenerlos seguros y mostrar para qué se utilizan.

Es por ello que la fundación debe garantizar la seguridad de los datos una de las formas es tener una política de los despliegues manteniendo los ambientes de producción y pruebas totalmente separados.

Garantizarle al usuario que los datos suministrados solo serán usados con fines de para la fundación y solo para realizar las adopciones.

Los datos ingresados al realizar los pagos en línea son realizados por un tercero llamado Mercado Pago quien tiene un gran historial de funcionamiento y experiencia en pagos de esta forma la ampliación no puede captar estos datos en ningún momento.

Pruebas funcionales

Para confirmar el funcionamiento de la aplicación se realizan pruebas que permiten garantizar la calidad del producto, se utilizan pruebas de validación, que son actividades que son observadas por los usuarios finales, y esta verificación se da cuando el programa funciona como espera el usuario. (Pressman, 2010).

Prueba de validación por parte de los usuarios

Esta prueba se realiza poniendo a disposición un ambiente de pruebas para los colaboradores de la fundación y prueba el aplicativo y funcionamiento.

Para llevar a cabo la prueba se ejecutó el aplicativo en un equipo local, se le dio hizo entrega del equipo donde se levantó el servidor local, y algunos de los colaboradores de la fundación pudieron hacer uso de él. En total 7 persona pudieron hacer uso del sistema, luego de hacer uso del sistema, se les pidió a los colaboradores que respondieran una encuesta realizada en Google Form el cual valida el funcionamiento y el uso de la aplicación.

La primera pregunta realizada a los usuarios fue:

¿Resultado de gran utilidad el sistema para el registro de la información recolectada de los animales de la fundación?

7 respuestas

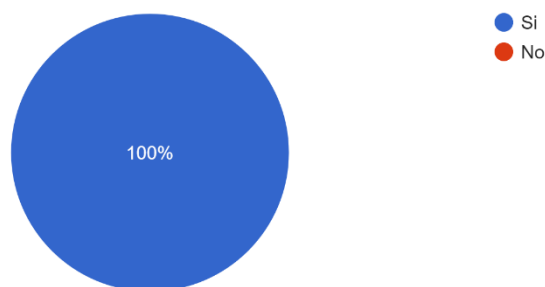


Ilustración 99 – Primer resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la primera pregunta fueron de 100% que es de utilidad la recolección de información de los animales de la fundación (ver Ilustración 89).

La segunda pregunta realizada fue:

¿Tuvo inconvenientes al realizar el registro de un nuevo animal?

7 respuestas

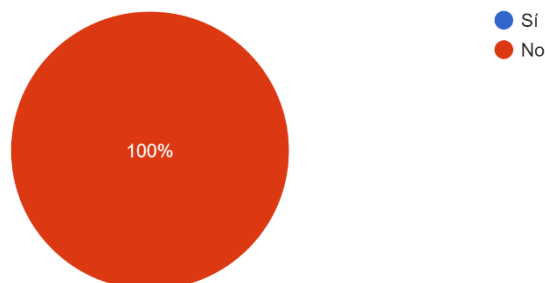


Ilustración 100- Segundo resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la segunda pregunta fueron de 100% que no tuvieron inconvenientes al registrar un nuevo animal (ver Ilustración 90) .

La tercera pregunta realizada a los usuarios fue:

¿Fue fácil usar la aplicación?

7 respuestas

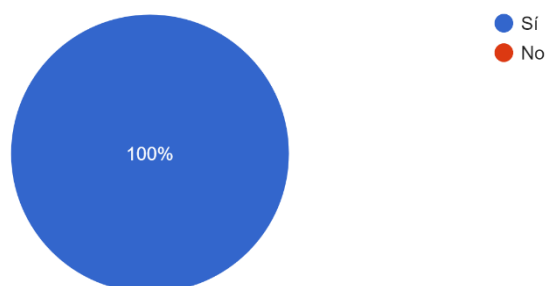


Ilustración 101 - Tercer resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la tercera pregunta fueron que al 100% de los colaboradores les resultó fácil usar la aplicación (ver Ilustración 91).

La cuarta pregunta realizada a los usuarios fue:

¿Tuvo inconvenientes al momento de registrar la información en los diferentes formularios?

7 respuestas

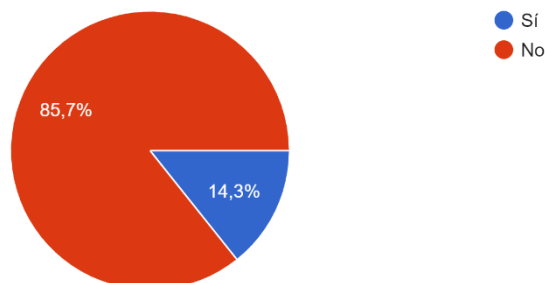


Ilustración 102 – Cuarto resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la cuarta pregunta fueron que al 85.7% de los colaboradores no tuvieron inconvenientes en registrar datos en los diferentes formularios existentes (ver 92Ilustración 101).

La quinta pregunta realizada a los usuarios fue:

¿La información que se visualiza en las tablas es la misma que ha registrada anteriormente?

7 respuestas

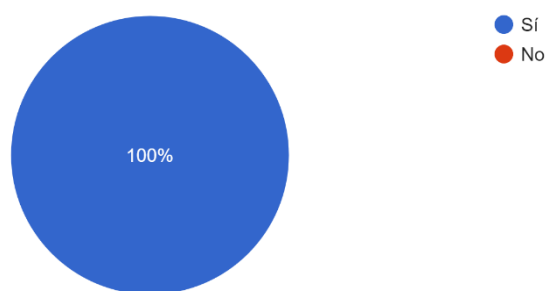


Ilustración 103 – Quinto resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la quinta pregunta fueron que al 100% de los colaboradores estuvieron de acuerdo que la información mostrada en las diferentes tablas concordaba con la registrada anteriormente (ver Ilustración 93).

La sexta pregunta realizada a los usuarios fue:

¿Es de utilidad que la aplicación permita almacenar fotos de los animales?

7 respuestas

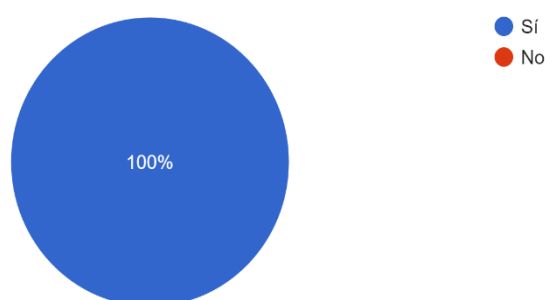


Ilustración 104 - Sexto resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la sexta pregunta fueron que al 100% de los colaboradores estuvieron de acuerdo que la aplicación permita almacena la foto de los animales es de bastante utilidad (ver Ilustración 94).

La séptima pregunta realizada a los usuarios fue:

¿Qué tipo de medio de pago le intereso más?

7 respuestas

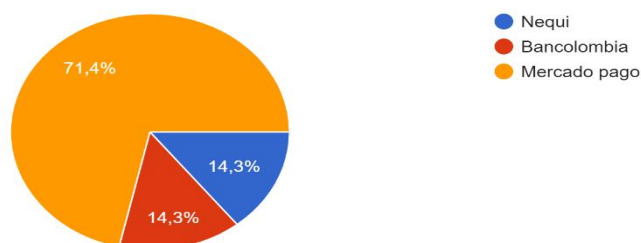


Ilustración 105 - Séptimo resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la séptima preguntan fueron que al 71.4% de los colaboradores tuvieron un mayor interés por e medio de pago en línea mercado pago (ver Ilustración 95).

La octava pregunta realizada a los usuarios fue:

¿Pudo ver la información registrada de las solicitudes de adopción?
7 respuestas

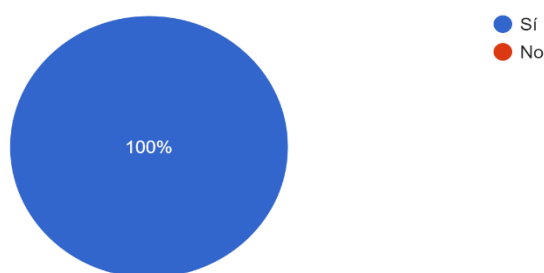


Ilustración 106 - Octavo resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la octava preguntan fueron que el 100% de los colaboradores pudieron ver la información de las solicitudes de adopción (ver Ilustración 96).

La novena pregunta realizada a los usuarios fue:

¿Tuvo inconvenientes al momento de cambiar la información de su perfil?
7 respuestas

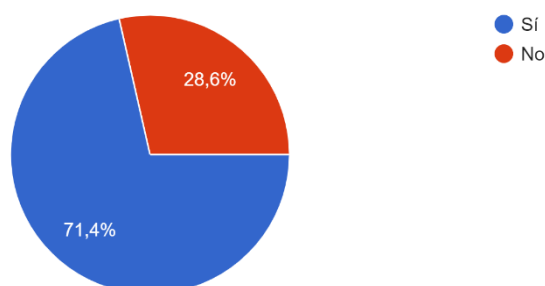


Ilustración 107 - Noveno resultado de la encuesta de aceptación de la aplicación

Fuente: Elaboración propia

Los resultados que se obtuvieron en la novena pregunta fueron que el 71.4% de los colaboradores no tuvieron inconvenientes al editar su información de perfil (ver Ilustración 97).

Conclusiones

Este proyecto fue desarrollado utilizando nuevas tecnologías, tales como con Golang y React, lo cual representó para mí la oportunidad de apropiarme de herramientas de gran importancia, que me han permitido crecer y evolucionar en mi vida profesional. Esto incluye el empleo de la base de datos documental mongoDB, con la que pude incursionar en otro tipo de modelo físico de datos, que resulta ser muy eficiente cuando se trata de este tipo de aplicaciones web.

Gracias a la metodología Kanban, se logró una mayor organización con respecto a las actividades planificadas para el desarrollo del proyecto, ya que esta metodología permite la gestión del tiempo y la reprogramación de actividades en caso de falla.

De un análisis de requisitos concienzudo y detallado se obtuvo una identificación exacta de las necesidades de la fundación con respecto al registro, apadrinamiento y adopción de mascotas, proveyendo de una base sólida para el diseño y construcción de la aplicación web que facilitará estos procesos de manera eficaz.

El sistema de información, producto de este trabajo de grado, fue validado con el personal con la Fundación Jhoana Sánchez en lo concerniente al registro de información de animales y adopciones, obteniendo como resultado, que fue de fácil manejo, con interfaces sencillas y con un buen diseño.

Recomendaciones

- Se recomienda a la fundación continuar con el desarrollo del sistema de información para su mejora y actualización, pues es de gran impacto tanto para los animales como para la salud pública.
- Se recomienda agregar pruebas de unidad automatizadas tanto en el front-end como para en back-end esto garantizará que al realizar futuros cambios la integridad de los componentes no se vea afectada.
- Se recomienda agregar una política que garantice el respaldo y recuperación de la base de datos.
- Se recomienda crear un servicio independiente para el almacenamiento de las imágenes.

Bibliografía

Álvarez, F. (05 de Octubre de 2016). *www.24horas.cl*. Obtenido de 24horas:

<https://www.24horas.cl/tendencias/mascotas/descubre-la-importancia-de-las-mascotas-en-la-vida-humana-2153400>

Cabrera Paucar, S. N. (2017). *Desarrollo de una aplicación web de adopción de mascotas en la ciudad de Loja*. Loja.

Cohn, M. (2009). *User Stories Applied*. Addison-Wesley.

funcionpublica.gov.co. (30 de 11 de 2022). *www.funcionpublica.gov.co*. Obtenido de

<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981>

Fundación Aquae. (2022). *Fundación Aquae*. Obtenido de

<https://www.fundacionaquae.org/la-inteligencia-artificial-y-el-test-de-turing-2/#:~:text=La%20inteligencia%20artificial%20es%20una,de%20una%20serie%20de%20datos.>

Go. (2022). *Golang*. Obtenido de <https://golang.org/doc/>

Golearn. (2022). *Golearn*. Obtenido de <https://github.com/sjwhitworth/golearn>

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. (2014).

Metodología de la investigación. Mc Graw Hill.

IONOS. (2019). *Digital Guide IONOS*. Obtenido de

<https://www.ionos.es/digitalguide/servidores/know-how/golang/>

Iris. (2022). *Iris*. Obtenido de <https://github.com/kataras/iris>

Iris. (2022). *Iris*. Obtenido de <https://docs.iris-go.com/iris/about>

Java Point. (2022). *www.javatpoint.com*. Obtenido de [https://www.javatpoint.com/k-](https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

[nearest-neighbor-algorithm-for-machine-learning](https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

JWT token. (2022). Obtenido de <https://jwt.io/>

Kanban Tool. (s.f.). *www.kanbantool.com*. Obtenido de

<https://kanbantool.com/es/metodologia-kanban>

Material Design. (2022). *Material Design*. Obtenido de <https://m3.material.io/>

Méndez Melo, A. C., Villafañe Castaño, A., & Martínez España, J. &. (2019).

Aplicación móvil para adopción de mascotas abandonadas

“peluditos.com.” (Vol. 1, Issue 1). UNAD.

Mercado Pago. (2022). *Mercado pago*. Obtenido de

<https://sandbox.mercadopago.com.co>

Mintic. (15 de Enero de 2022). *Mintic*. Obtenido de

https://normograma.mintic.gov.co/mintic/docs/resolucion_mintic_0512_2019.htm

MongoDB. (2022). *MongoDB*. Obtenido de <https://www.mongodb.com/>

Pressman, R. s. (2010). *Ingenieria de software Un enfoque práctico*. Mexico:

McGRAW-HILL INTERAMERICANA.

React JS. (2022). Obtenido de <https://reactjs.org/>

Reactive Programming. (2022). *Reactive Programming*. Obtenido de

<https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/rest>

Toapanta Gualpa, M. I. (2019). *Desarrollo de un sistema de gestión sectorizado de adopción de mascotas para la fundación camino a casa*. Quito.

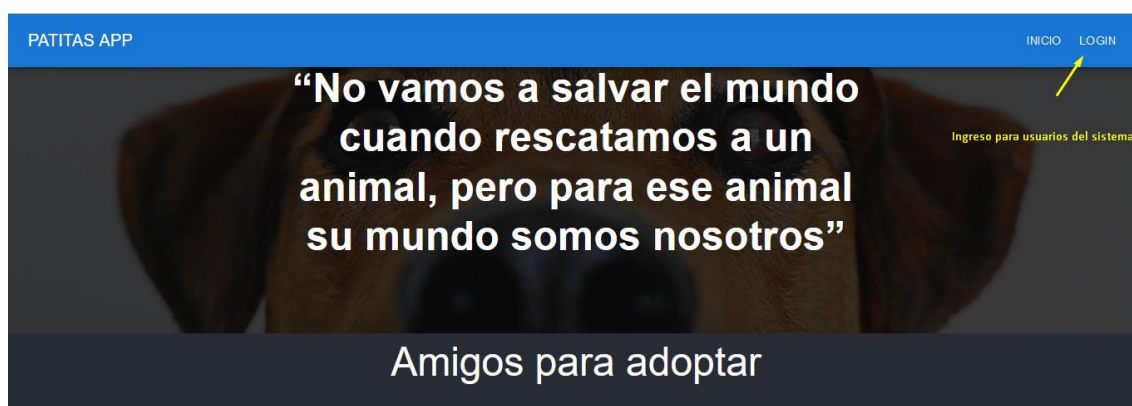
Zaldumbide Burbano, V. N. (2018). *Desarrollo de una aplicación web de red social para estimular los vínculos afectivos entre mascotas y sus dueños (Vol. 2)*.

Quito.

Zorita Bote, I. (2016). *Aplicación de gestión de mascotas OURPET*. Universidad Autonoma de Madrid.

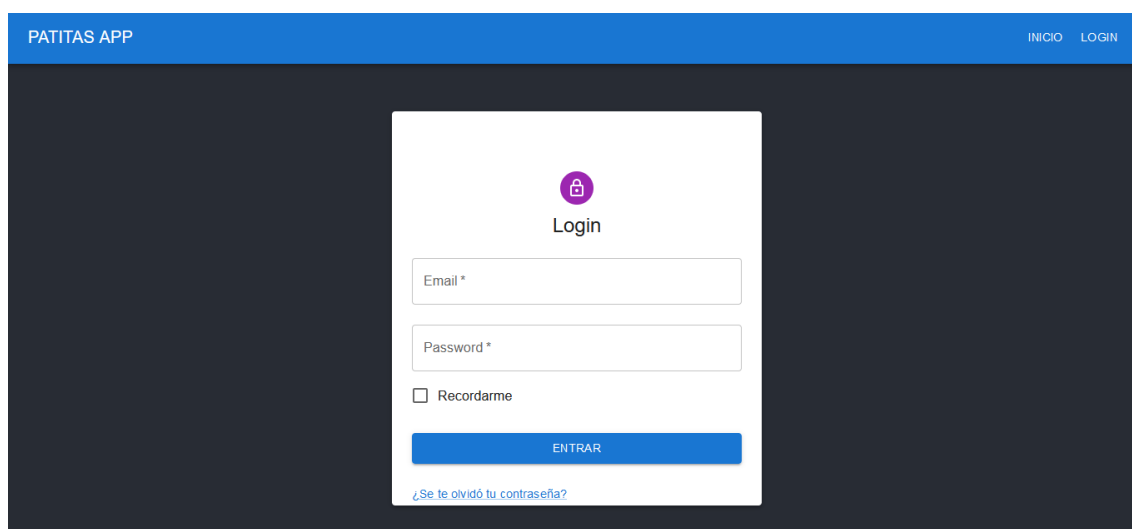
Anexo a. Manual del usuario

1. Inicio de Sesión



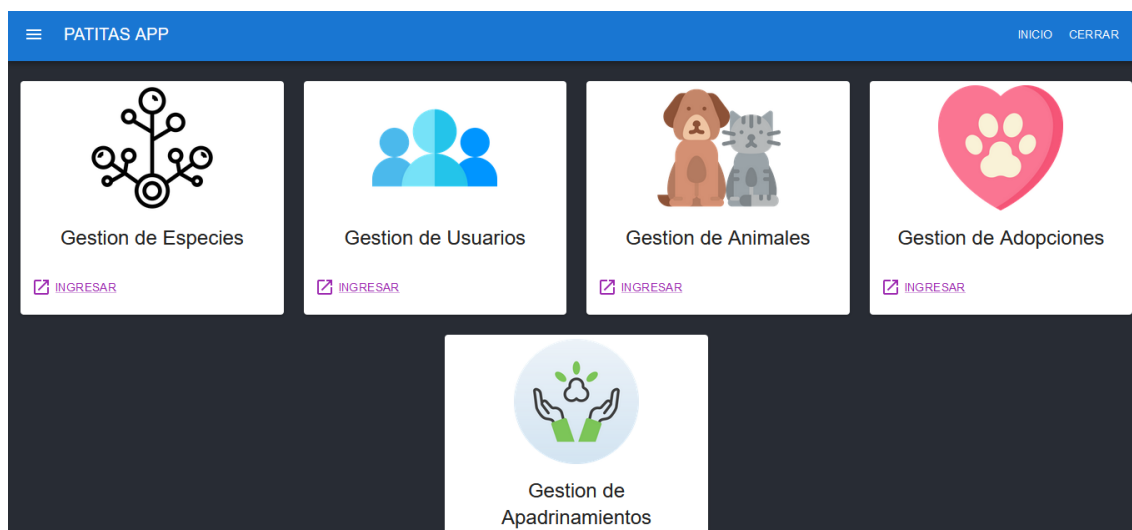
Página de inicio

Al ingresar en la página de la fundación se puede visualizar en el menú la opción de login para los usuarios administradores que los lleva a la página de login donde ingresarán correo y contraseña para acceder al sistema.



Página de inicio

Luego de ingresar su correo y contraseña y que estos datos sean correctos, el usuario ingresa al panel administrativo donde tiene disponibles las diferentes opciones que puede realizar.



Página de inicio

2. Gestión de animales

El usuario ingresa en este módulo para la gestión de los diferentes animales que son ingresados en la fundación. Inicialmente, el sistema presenta una listado donde se puede seleccionar el animal y dando clic en botón de Ver se pueden consultar sus datos en forma detallada.

PATITAS APP										INICIO	CERRAR
Animales										Acción para agregar animales → + Agregar	
↑ Nombre	Color	Tamaño	Esterilizado	En Adopción	Fecha Nacimiento	Especie	Sexo	Fecha Creación	Fecha de actualización		
Tarzan	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER	
Turquesa	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER	
dana	blanco	mediano	Esterilizado	NO	31/12/0001	Perro	F	30/10/2022	30/10/2022	VER	
dana	blanco	mediano	Esterilizado	NO	13/10/2015	Perro	F	30/10/2022	30/10/2022	VER	
docky	negro	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER	
mono	marron claro	mediano	sin Esterilizar	NO	31/12/0001	Perro	M	30/10/2022	30/10/2022	VER	
peria	blanco	pequeño	sin Esterilizar	NO	10/10/2016	perro	F	31/12/0001	31/12/0001	VER	
Rows per page: 10 1-7 of 7 < >											

Página de lista de animales

3. Crear animal

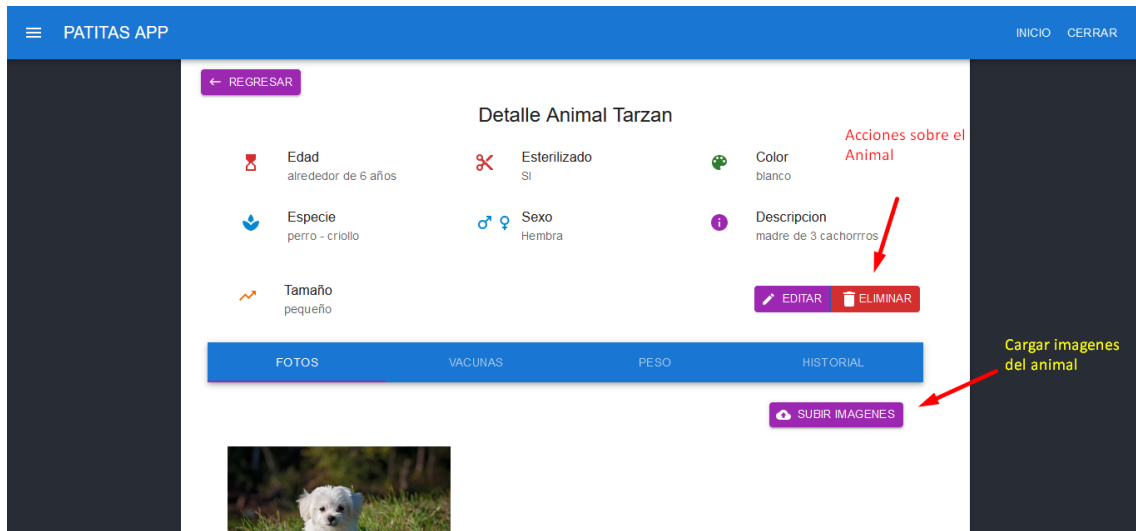
En esta pantalla se agrega la información del animal como: nombre, color, tamaño, fecha de nacimiento, especie, raza, sexo y si está esterilizado. Una vez se han completado los datos se activa el botón de enviar para hacer el respectivo registro.

PATITAS APP										INICIO	CERRAR								
← REGRESAR																			
Formulario para el registro de Animales																			
Nombre de Animal *				Color de Animal *															
vainla				blanco															
Tamaño de Animal *				Fecha de Nacimiento															
pequeño				01/12/2005															
Especie del Animal *				Raza del Animal *															
Perro				Criollo															
Sexo																			
☒ Hembra ☐ Macho ☐ Esterilizado																			
ENVIAR																			

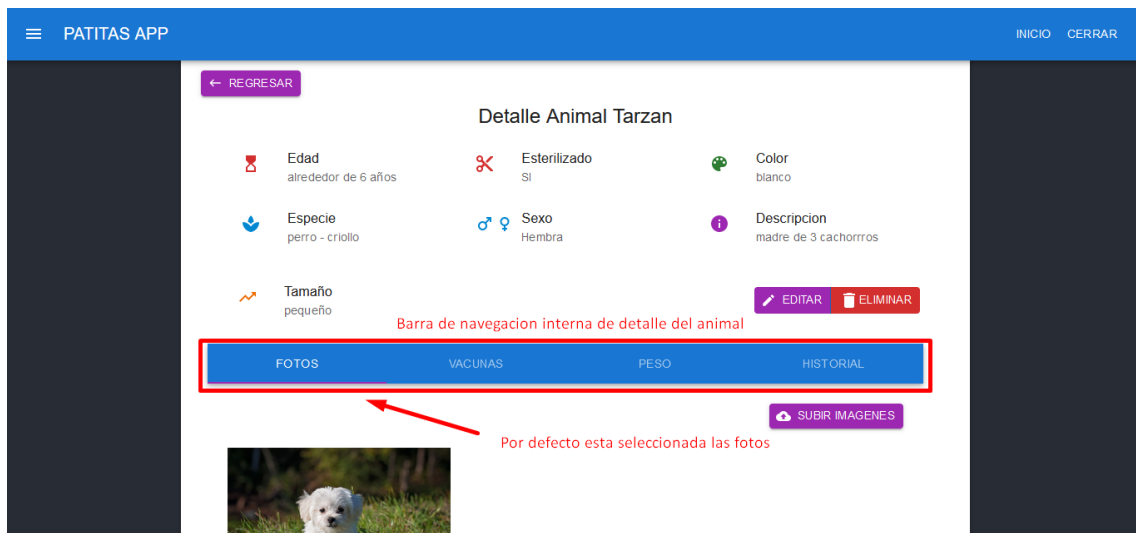
Formulario de Crear animal

4. Detalle del animal

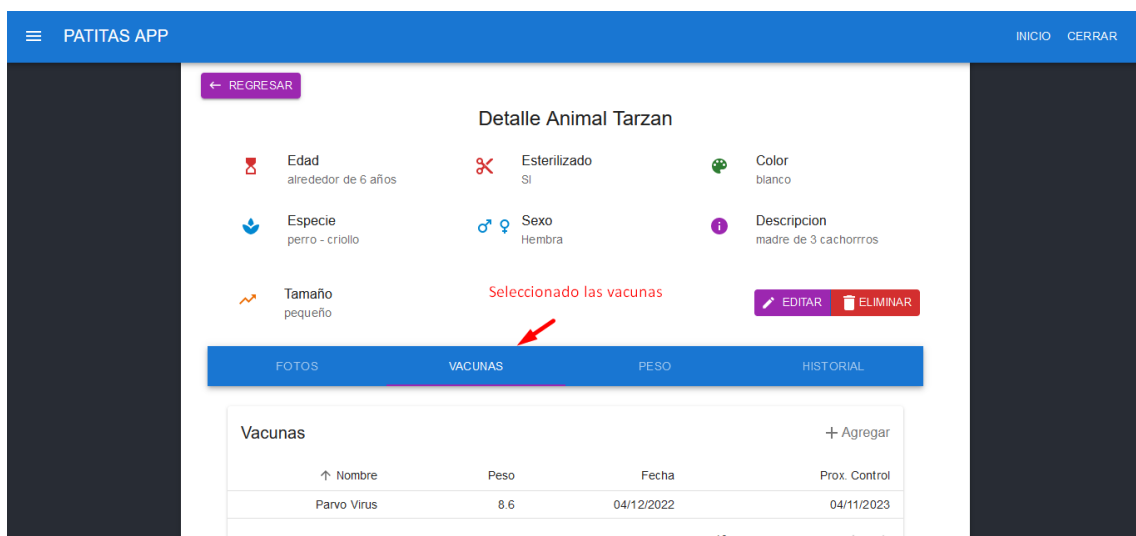
Esta pantalla permite visualizar la información del animal y navegar entre los diferentes tipos de información foto, vacunas, control de peso e historial a través de la barra de opciones que se encuentra en el interior de esta interfaz



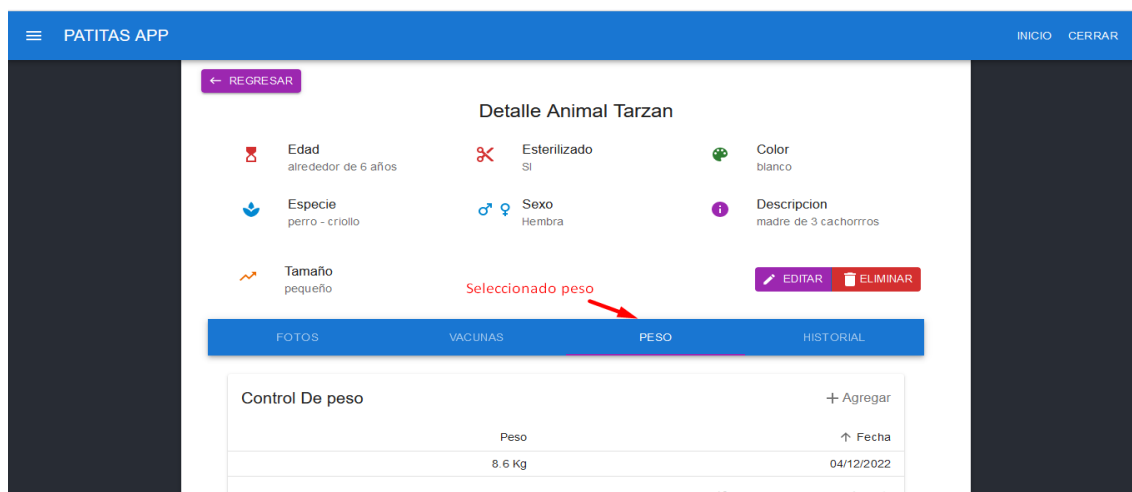
Detalle del animal



Detalle del animal-Fotos



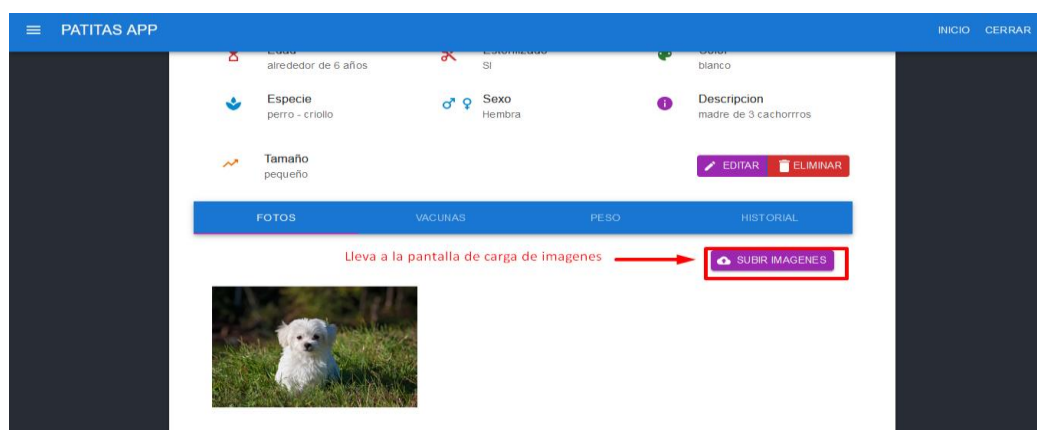
Detalle del animal-Vacunas



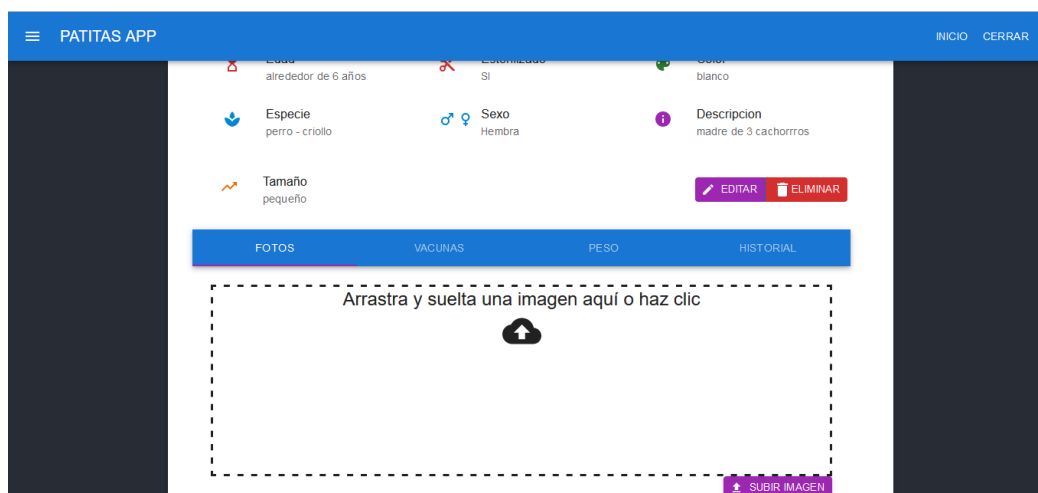
Detalle del animal-Peso

5. Cargar fotos del animal

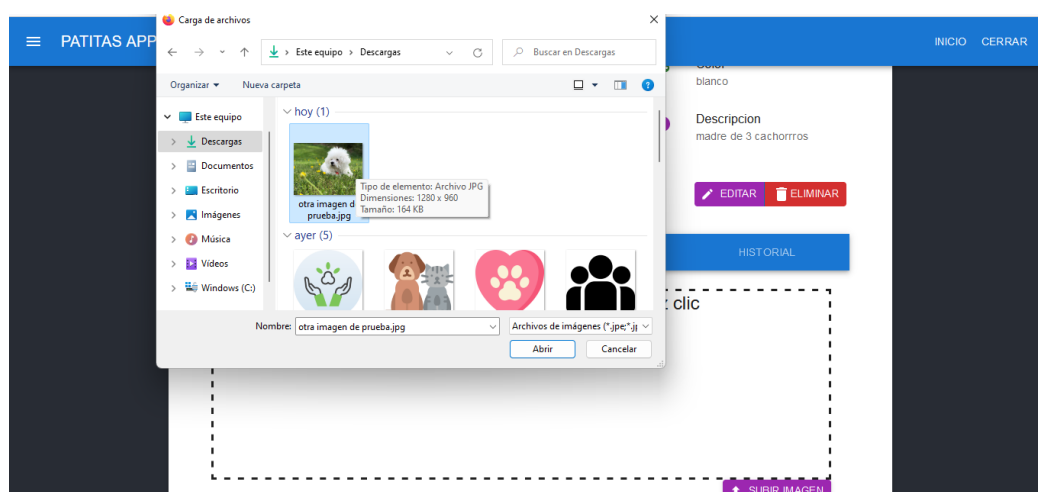
Esta funcionalidad permite agregar imágenes a los animales con el fin de que estas sean publicadas y de esta forma puedan ser seleccionados para una futura adopción.



Detalle del animal - Foto ir a carga de imágenes



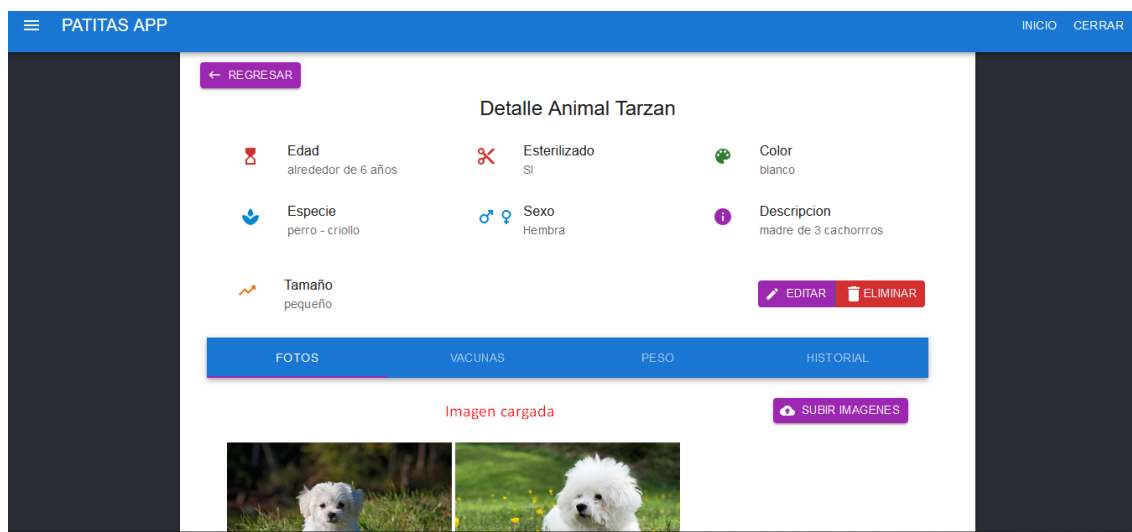
Detalle del animal - Carga de imágenes



Detalle del animal – Selección de la imagen a cargar



Detalle del animal – Previsualización de la imagen



Detalle del animal – imagen cargada con éxito

6. Editar animal

En esta pantalla se edita la información del animal como: nombre, color, tamaño, fecha de nacimiento, especie, raza, sexo y si está esterilizado. Una vez se han completado los datos, se activa el botón de enviar para realizar la actualización de sus datos.

Editar del animal

Ver solicitudes de adopción

En esta pantalla los usuarios de la fundación pueden ingresar para listar los animales que tienen solicitudes de adopción. Al dar clic en el botón de ver, se pueden apreciar los datos de la persona que ha enviado una solicitud de adopción. En la última

columna de cada fila hay un botón de selección, con el que se puede cambiar el estado de la solicitud dependiendo del criterio de la fundación.

PATITAS APP								INICIO	CERRAR
Solicitudes de adopción									
↑ Nombre	Color	Tamaño	Esterilizado	En Adopción	Fecha Nacimiento	Especie	Sexo		
mono	marron claro	mediano	sin Esterilizar	SI	31/12/0001	Perro	M		VER
perla	blanco	pequeño	sin Esterilizar	SI	10/10/2016	perro	F		VER
								Rows per page: 10 ▾	1-2 of 2 < >

Lista de animales con solicitudes de adopción

PATITAS APP

INICIO CERRAR

← REGRESAR

Animal mono

Edad

casi 2022 años

Esterilizado

SI

Color

marron claro

Tamaño

mediano



Sexo

Macho



Descripción

ELIMINAR SOLICITUDES

Solicitudes de adoptantes

↑ Nombres	Apellidos	Documento	Dirección	Teléfono	Email	Estado		
camila	contreras	CC-565656	cucuta-ciudad jardin-56 vale ve	565656	blade_liger@hotmail.com	inicial	31/12/1969	Cambiar Estad
enyerson	camero	CC-1092343	cucuta-bocono-av siempre viva	565656565	blade_liger@hotmail.com		31/12/1969	Cambiar Estad
enerson	camero	CC-1092343	cucuta-bocono-av	565656565	blade_liger@hotmail.com		31/12/1969	Cambiar Estad

Detalle de la solicitud de adopción

PATITAS APP

INICIO CERRAR

← REGRESAR

Animal mono

Edad

casi 2022 años

Esterilizado

Si

Color

marron claro

Tamaño

mediano

Sexo

Macho

Descripcion

ELIMINAR SOLICITUDES

Solicitudes de adoptantes

Estado de las solicitudes de adopción

↑ Nombres	Apellidos	Documento	Dirección	Teléfono	Email	Estado
camila	contreras	CC-565656	cucuta-ciudad jardin-56 vale ve	565656	blade_tiger@hotmail.com	inicial 31/12/1969
enyerion	camero	CC-1092343	cucuta-bocono-av siempre viva	565656565	blade_tiger@hotmail.com	31/12/1969
enyerion	camero	CC-1092343	cucuta-bocono-av	565656565	blade_tiger@hotmail.com	31/12/1969

Seleccione

Aprobar

en Espera

Pendiente

Rechazar

Cambiar estado de la solicitud de adopción