INGENIERIA DE SISTEMAS

DESARROLLO DE UN SISTEMA WEB CON ASISTENCIA DE CHATBOT PARA LA GESTIÓN ADMINISTRATIVA DEL REFUGIO "NARICES FRÍAS"

Cochabamba - Bolivia
2025

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO I

INTRODUCCIÓN

CAPÍTULO I

INTRODUCCIÓN

1. INTRODUCCION

En la actualidad, la gestión eficiente de la información es clave para mejorar los procesos en distintas organizaciones, incluyendo los refugios de animales. La adopción responsable y el bienestar de los animales dependen en gran medida de un registro adecuado de datos, un seguimiento posterior a la adopción y la disponibilidad de recursos para su cuidado

El desarrollo de soluciones tecnológicas especializadas se ha convertido en una herramienta fundamental para optimizar la administración de estos refugios, asegurando un control preciso sobre las adopciones, mejorando la transparencia en la gestión de donaciones y facilitando la comunicación entre el refugio y la comunidad

1.1 ANTECEDENTES

Se estima que en el mundo hay más de 600 millones de animales de compañía sin hogar. Una cifra alarmante pero que cada año sensibiliza más a la población. Si bien es cierto que en algunas partes del mundo se ha tomado conciencia, aún existe un gran porcentaje que demuestra ser indiferente ante esta situación. (Axonvet, 2023).

Según un estudio de la Fundación, las principales causas de abandono de mascotas incluyen camadas no deseadas (15%), pérdida de interés por el animal (13%), problemas de comportamiento (12%), fin de temporada de caza (11%), factores económicos (10%), cambio de domicilio (9%), alergias (5%), ingreso a un centro hospitalario (5%), falta de tiempo o espacio (4%), nacimiento de un hijo o hija (4%), divorcio (3%), vacaciones (2%) y pérdida de empleo (2%). (Fundacion affinity, 2024).

En el contexto boliviano, la población canina es significativamente alta. Según Paredes (2023), se estima que hay 2.938.458 perros en el país, lo que equivale a un perro por cada tres habitantes. Esta cifra supera ampliamente la recomendación

de la Organización Mundial de la Salud (OMS), que sugiere un perro por cada 10 habitantes. Además, las condiciones de tenencia responsable suelen ser irregulares.

En los municipios de Santa Cruz, Cochabamba, La Paz, El Alto, Tarija y Sucre existen leyes locales sobre la tenencia de animales domésticos, pero son deficientes en cuanto al registro de mascotas y su difusión. Además, muchos propietarios desconocen qué está permitido y qué no, y las sanciones por abandono son mínimas, lo que agrava los problemas de salud pública. (Quezada,2024)

“Narices frías” es un refugio que desde hace años viene realizando una labor destacable en la sociedad cochabambina con los perritos callejeros, un problema serio que se agudiza en nuestra ciudad debido a la falta de conciencia en el cuidado de los animales domésticos.

El refugio también pide la colaboración de otros ciudadanos o empresas comprometidas con los perritos para realizar eventos u otras actividades para tener una ayuda económica y así también recibiendo donaciones en estos eventos.

1.2 PLANTEAMIENTO DEL PROBLEMA

Actualmente, el refugio “Narices Frías” opera con registros en papel lo que genera problemas de organización, pérdida de información y dificultades en la toma de decisiones.

Uno de los problemas más críticos es el proceso de adopción, ya que no existe un sistema automatizado que facilite la postulación de adoptantes ni el seguimiento de los animales después de la adopción. El administrador debe revisar manualmente cada solicitud y comunicarse con los interesados de forma individual, lo que puede generar retrasos y confusión en la gestión. Además, no hay un registro estructurado de los animales adoptados y su estado después de la adopción, lo que dificulta la detección de posibles casos de maltrato o abandono.

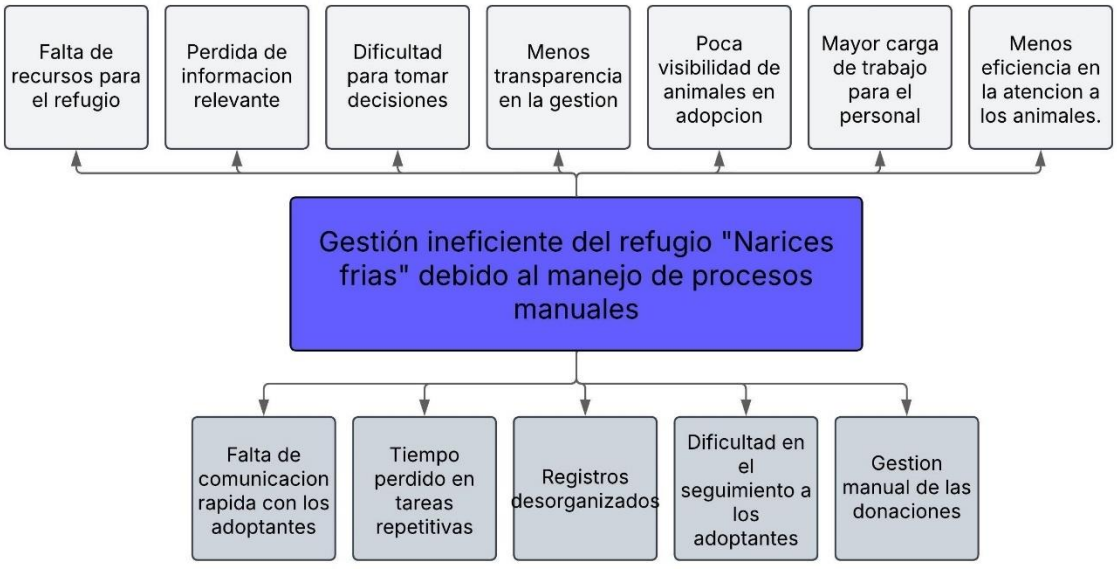
1.2.1 Situación problemática y/o requerimiento de la institución

Desde su fundación el refugio “Narices Frías” maneja sus adopciones, registros de animales, registros de adoptantes, donaciones y seguimientos en formato físico, pero con el paso del tiempo y el creciente registro de animales hace que el proceso sea deficiente ya que a veces suelen existir errores al momento de registrar la adopción de un animal.

Existen varios factores que ocasionan errores al momento de dar en adopción, por ejemplo, al guardar los datos a veces se escribe mal el nombre del animal o del adoptante entonces se guarda de manera errónea y al momento de buscar no es el verdadero dato que se debería haber guardado y también esta manera de guardar datos crea demora en el proceso de adopción.

Además, la cantidad de registros va creciendo con el paso del tiempo y ha provocado acumulación de varios cuadernos a lo largo del tiempo y esto causa que se pierda mucho tiempo en la búsqueda de información específica de una adopción en los registros. Por otro lado, existe el riesgo de pérdida porque son datos físicos.

ILUSTRACIÓN N° 1: Árbol de problemas



FUENTE: Elaboración propia.

1.2.2 Objeto de estudio

El objeto de estudio de este proyecto es la digitalización y optimización de la gestión del refugio de animales "Narices Frías", con el fin de mejorar los procesos de adopción, registro de animales, administración de donaciones y seguimiento post-adopción.

Este estudio abarca tanto los aspectos técnicos (requerimientos, diseño, desarrollo del sistema) como los operativos y administrativos del refugio, con el objetivo de crear una herramienta que optimice sus procesos y garantice un mejor bienestar para los animales.

1.2.3 Estudio de soluciones

El objeto de estudio es la digitalización y optimización de la gestión del refugio "Narices Frías" a través del desarrollo de un sistema web que centralice y gestione la información de manera eficiente

TABLA 1: VALORACIÓN DE SOLUCIONES

Casos de estudio	Problema	Solución aplicada	Valoración
Refugio "Amigos Peludos", Madrid, España. (ADLP.com, 2023)	Dificultad para comunicarse con adoptantes y responder preguntas frecuentes.	Implementación de un chatbot en la página web del refugio para responder preguntas frecuentes y proporcionar información sobre adopciones.	Solución efectiva. Redujo la carga de trabajo del personal y mejoró la comunicación con los adoptantes.
Refugio "Huellitas", Buenos Aires, Argentina. (huellitasperdidasok, 2025).	Dificultad para gestionar donaciones y falta de transparencia en el uso de fondos.	implementación de una plataforma de donaciones en línea con seguimiento detallado de transacciones y reportes de uso de fondos.	Solución muy efectiva. Aumentó la transparencia y la confianza de los donantes, facilitando la recaudación de fondos.

FUENTE: Elaboración propia.

1.2.4 Pregunta de investigación

¿Cómo el Desarrollo de un sistema web con asistencia de chatbot mejorará la gestión administrativa del refugio "Narices Frías”

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 Objetivo General

Desarrollar un sistema web con asistencia de chatbot para la gestión administrativa del refugio "Narices Frías"

1.3.2 Objetivos específicos

- Analizar los requerimientos para identificar las funcionalidades de la gestión del refugio de animales "Narices Frías".
- Elaborar el diseño de la base de datos, estructurando la información de animales, adopciones, usuarios, lista negra, donaciones y seguimientos para un almacenamiento eficiente.
- Diseñar la interfaz de usuario asegurando una experiencia intuitiva y accesible para los adoptantes y el administrador del refugio.
- Construir la lógica del servidor, implementando los controladores, modelos y API necesarias para gestionar la información del sistema de manera eficiente.
- Implementar un chatbot inteligente que brinde respuestas automáticas a preguntas frecuentes de los adoptantes y facilite la comunicación con el refugio las 24 horas.
- Realizar pruebas funcionales y de rendimiento para garantizar la estabilidad, seguridad y correcto funcionamiento del sistema web.

1.4 DEFINICIÓN DE VARIABLES

TABLA 2: VARIABLES DE LA INVESTIGACIÓN

<table>
<tr><td colspan="6">Pregunta de investigación
¿Cómo el Desarrollo de un sistema web con asistencia de chatbot mejorará la gestión administrativa del refugio "Narices Frías"</td></tr>
<tr><td colspan="6">Objetivo general
Desarrollar un sistema web con asistencia de chatbot para la gestión administrativa del refugio "Narices Frías"</td></tr>
<tr><td>Variable</td><td>Definición conceptual</td><td>Definición operacional</td><td colspan="2">Indicadores (qué se espera notar u obtener de esta variable)</td><td>Herramientas (Instrumentos)</td></tr>
<tr><td rowspan="3">Variable Independiente

Desarrollo de un sistema web con asistencia de chatbot</td><td rowspan="3">“especifica qué actividades u operaciones deben realizarse para medir una variable. Una definición operacional nos dice que para recoger datos respecto de una variable, hay que hacer esto y esto otro,” (HERNANDEZ, FERNANDEZ, & BAPTISTA, 2010, pág. 111)</td><td rowspan="3">Un sistema web es una plataforma digital accesible desde un navegador que permite gestionar y automatizar procesos. Un chatbot es un programa de inteligencia artificial que interactúa con los usuarios de forma automática, brindando asistencia en tiempo real. Su integración mejora la eficiencia administrativa y la experiencia del usuario</td><td>I1. Nivel de automatización de procesos administrativos</td><td>$\frac{Cantidad\ de\ tareas\ automatizadas}{Total\ de\ tareas\ administrativas\ del\ refugio}$</td><td>Análisis de flujo de trabajo antes y después del sistema, entrevistas con el administrador.</td></tr>
<tr><td>I2. Reducción del tiempo en la gestión de adopciones</td><td>Tiempo promedio para procesar una adopción antes y después del sistema</td><td>Mediciones de tiempo de respuesta, entrevistas a usuarios y administrador.</td></tr>
<tr><td>I3. Interacción y satisfacción de los usuarios con el chatbot</td><td>$\frac{Número\ de\ interacciones\ exitosas}{Total\ de\ interacciones\ con\ el\ chatbot}$</td><td>Análisis de registros del chatbot.</td></tr>
</table>

Pregunta de investigación ¿Cómo el Desarrollo de un sistema web con asistencia de chatbot mejorará la gestión administrativa del refugio "Narices Frías"					
Objetivo general Desarrollar un sistema web con asistencia de chatbot para la gestión administrativa del refugio "Narices Frías"					
Variable	**Definición conceptual**	**Definición operacional**	**Indicadores (qué se espera notar u obtener de esta variable)**		**Herramientas (Instrumentos)**
Variable Dependiente Mejora en la gestión administrativa del refugio "Narices Frías"	Optimización de los procesos administrativos mediante la digitalización y automatización, reduciendo la carga de trabajo manual y mejorando la eficiencia operativa del refugio.	Evaluación del impacto del sistema en la administración del refugio, comparando los procesos antes y después de su implementación	I1. Reducción de errores en el registro de datos	Cantidad de errores antes y después del sistema	Revisión de registros y comparación de datos antes y después de la digitalización.
			I2. Satisfacción del administrador con el nuevo sistema	Nivel de satisfacción según entrevistas	Entrevistas al administrador del refugio.
			I3. Aumento en la cantidad de adopciones procesadas	Número de adopciones antes y después del sistema	Análisis de registros de adopciones y comparación de datos históricos.
			I4. Reducción del uso de papel en la gestión del refugio	Cantidad de documentos físicos utilizados antes y después del sistema	Revisión y medición del uso de papel antes y después.

FUENTE: Elaboración propia

1.5 DELIMITACIÓN

TABLA 3: DELIMITACIÓN DEL PROYECTO

Límites	Justificación
¿Hasta dónde se quiere llegar con este proyecto de grado?	El objetivo es desarrollar un sistema web funcional con un chatbot integrado que mejore la gestión administrativa del refugio "Narices Frías". Se espera que el sistema automatice tareas clave, como el registro de animales, el seguimiento de adopciones y la gestión de donaciones, mejorando la eficiencia y reduciendo la carga de trabajo manual del administrador.
¿Qué cosas pretende solucionar este proyecto?	Este proyecto pretende solucionar lo siguiente: Reducir el uso de papel y digitalizar la gestión del refugio, agilizar el proceso de adopción y seguimiento de los animales, mejorar la comunicación con los adoptantes mediante un chatbot que responda preguntas frecuentes y facilitar la gestión de donaciones y el control de los animales en el refugio.
¿Se pondrá a prueba el proyecto o sólo se ejecutará una experiencia piloto o no es necesario ninguna?	Este proyecto se pondrá a prueba

FUENTE: Elaboración propia en base a delimitaciones del proyecto

1.5.1 Límite temporal

Se proyecta que el proceso de desarrollo del sistema Web para este proyecto tenga una duración de seis meses, abarcando de febrero hasta julio de 2025. Durante este período, se llevarán a cabo múltiples etapas, incluyendo la planificación inicial, el diseño, la implementación, las pruebas y la puesta en marcha.

1.5.2 Límite geográfico

El sistema web estará disponible para toda la ciudad de Cochabamba, permitiendo que cualquier persona interesada en adoptar un animal pueda acceder al sistema, revisar los perfiles de los animales y enviar una solicitud de adopción. Sin embargo, la gestión y administración de los registros de animales, adopciones, donaciones y seguimientos será responsabilidad del refugio “Narices Frías”, ubicado en Cochabamba.

1.6 JUSTIFICACIÓN

La implementación de este proyecto es importante para mejorar la gestión del refugio de animales "Narices Frías". Actualmente, los registros y procesos administrativos se manejan manualmente o en archivos dispersos, lo que genera pérdida de información, dificultad en el seguimiento de adopciones y problemas en la gestión de donaciones. Este sistema permitirá centralizar y digitalizar la información, optimizando la administración y garantizando un control más eficiente de los animales, adoptantes y recursos del refugio.

Además, la tecnología facilita la automatización de procesos, la reducción de costos operativos y el acceso rápido a la información, beneficiando tanto a los administradores del refugio como a los adoptantes. La implementación de un chatbot también hará más fácil la comunicación con los usuarios, proporcionando respuestas en tiempo real a preguntas frecuentes y agilizando los procesos de adopción.

1.6.1 Justificación técnica

El sistema de gestión será desarrollado con herramientas digitales modernas, permitiendo una plataforma flexible y escalable que se adapte a las necesidades específicas del refugio. Su diseño garantizará un acceso eficiente a la información y facilitará el monitoreo en tiempo real de los procesos administrativos, mejorando la organización y optimizando la gestión de adopciones, donaciones y registros de los animales.

1.6.2 Justificación económica

La implementación del sistema reducirá costos operativos al minimizar el uso de papel y disminuir el tiempo invertido en la gestión manual del refugio. Asimismo, la visibilidad en línea permitirá aumentar el alcance de campañas de adopción y donaciones, fortaleciendo la sostenibilidad del refugio. Al optimizar la administración de recursos y mejorar la eficiencia en los procesos, se logrará un impacto positivo en la operatividad y financiamiento de la institución.

1.6.3 Justificación social

Este proyecto impactará positivamente en la comunidad al promover la adopción responsable de animales y garantizar su bienestar a través de un seguimiento adecuado. Además, la implementación de un chatbot mejorará la comunicación con los adoptantes, brindando información clara y oportuna sobre los procesos de adopción. También se evitará que personas con antecedentes negativos vuelvan a adoptar, protegiendo así a los animales y fomentando una cultura de adopción responsable.

1.7 TIPOLOGÍA DE PROYECTOS

Es importante declarar los del proyecto justificándolos de manera técnica y/o teórica.

TABLA 4: ALCANCES DE LA INVESTIGACIÓN

Alcances	Justificación
Exploratorio La investigación exploratoria es un tipo de investigación utilizada para estudiar un problema que no está claramente definido, por lo que se lleva a cabo para comprenderlo mejor, pero sin proporcionar resultados concluyentes. (Villacis, 2019)	Dado que actualmente la gestión del refugio "Narices Frías" se realiza mediante registros físicos y hojas de cálculo dispersas, es necesario realizar una investigación exploratoria para comprender a fondo las necesidades y problemáticas en la administración de adopciones, donaciones y seguimiento de animales.
Explicativa "En este alcance de la investigación se busca una explicación y determinación de los fenómenos. En el contexto cuantitativo se pueden aplicar estudios de tipo predictivo en donde se pueda generar una manipulación intencionada de la variable independiente, pueden permitir comprobar hipótesis que expliquen el comportamiento de un determinado fenómeno."(Ramos, 2020)	El desarrollo del sistema para el refugio "Narices Frías" requiere una comprensión detallada de cómo la digitalización de los procesos influirá en la gestión de adopciones, donaciones y el seguimiento de los animales. A través de un enfoque explicativo, se analizarán los factores que afectan la administración actual y cómo la implementación del sistema puede optimizar estos procesos.

FUENTE: Elaboración propia.

1.8 TIPO Y ESTUDIO DE LA INVESTIGACIÓN

El presente estudio corresponde a un diseño **no experimental**, ya que no se manipulan variables deliberadamente, sino que se analiza la situación actual del refugio "Narices Frías" y se propone una solución para mejorar su gestión.

Asimismo, el estudio será de tipo **transversal**, ya que la recolección de datos se realizará en un solo momento para diagnosticar las problemáticas en la administración del refugio y evaluar cómo la digitalización puede optimizar sus procesos.

1.9 TÉCNICAS E INSTRUMENTOS DE INVESTIGACIÓN

TABLA 5: DECLARACIÓN DE USO DE INSTRUMENTOS ¡Error! Marcador no definido.

Tipo	Instrumento	A quién o a qué	Para qué
Teórico	Análisis documental	Informes y registros del refugio	Identificar problemas en la gestión actual.
	Mapa mental	Procesos de gestión del refugio	Visualizar las relaciones entre los procesos de adopción, donaciones y seguimiento de animales.
Empírico	Entrevista	Administrador del refugio	Obtener información sobre sus necesidades y problemas actuales.
	Entrevista	Administrador del refugio	Conocer la percepción y expectativas sobre el sistema digital.

FUENTE: Elaboración propia.

1.10 POBLACIÓN Y MUESTRA

En este estudio, la población está conformada por el personal administrativo del refugio "Narices Frías" y los cinco voluntarios que participarán en la validación y prueba del sistema.

Dado que el proyecto se enfocará exclusivamente en este refugio y se trabajará con la totalidad de las personas involucradas, no es necesario seleccionar una muestra representativa. En su lugar, se aplicará un censo, analizando la participación completa del equipo para garantizar que el sistema responda de manera óptima a sus necesidades específicas.

1.11 PLANIFICACIÓN DEL PROYECTO

1.11.1 Hoja de costos de ejecución del proyecto

El presupuesto total precisado se puede ver en la siguiente tabla.

TABLA 6: PRESUPUESTO DEL PROYECTO

Descripción	Cantidad	Unidad	Costo unitario (Bs.)	Costo total (Bs.)
Impresiones	1000	hojas	0,20	200,00
Empastados	5	unidades	40	200,00
CDs	2	unidades	12	24,00
Fotocopias	50	unidades	0,50	25,00
Transporte	8	cargas	40	320,00
Internet	4	meses	160	640,00
			TOTAL (Bs,)	**1409,00**

FUENTE: Elaboración propia en base a necesidades del proyecto

1.1.1. Diseño de Marcos

TABLA 7: DISEÑO DE MARCOS

Marco	Elementos del Marco	Comentario sobre la necesidad de ejecución de este marco para la realización del proyecto o tesis	Indicación de las fuentes a las que se recurrirá para la elaboración de este marco
2.1. Marco Histórico	**2.1.1. Evolución de la adopción de animales**	Es importante conocer cómo ha evolucionado la adopción de animales en refugios, las estrategias utilizadas en el pasado y su impacto en la sociedad.	Libros sobre historia del rescate y adopción de animales, informes de organizaciones de bienestar animal.
	2.1.2. Historia del refugio "Narices Frías"	Conocer la historia del refugio permite contextualizar el problema actual y justificar la necesidad de un sistema digitalizado.	Entrevistas con fundadores del refugio, documentos internos, registros previos.
	2.1.3. Métodos tradicionales de gestión en refugios	Analizar cómo se han gestionado históricamente los refugios permitirán identificar ineficiencias y la necesidad de digitalización.	Artículos sobre gestión de refugios, experiencias de otras organizaciones de bienestar animal.
2.2. Marco referencial	**2.2.1. Sistemas de gestión para refugios de animales**	Se analizarán otras plataformas similares para comprender qué características y funcionalidades pueden ser útiles.	Documentación de software de gestión de refugios, artículos académicos.
	2.2.2. Impacto de la digitalización en organizaciones sin fines de lucro	Se revisará cómo la adopción de herramientas digitales ha mejorado la gestión en otras organizaciones sin fines de lucro.	Estudios de caso, artículos académicos sobre transformación digital.

Marco	Elementos del Marco	Comentario sobre la necesidad de ejecución de este marco para la realización del proyecto o tesis	Indicación de las fuentes a las que se recurrirá para la elaboración de este marco
2.3. Marco Conceptual	**2.3.1. Adopción de animales**	Se definirá el concepto de adopción responsable, los requisitos y regulaciones que la rigen.	Normativas sobre adopción de animales, estudios sobre bienestar animal.
	2.3.2. Seguimiento post-adopción	Se explorará la importancia del seguimiento en la adopción y cómo afecta la tasa de éxito en adopciones responsables.	Informes de refugios, estudios de comportamiento animal post-adopción.
2.4. Marco teórico	**2.4.1. Modelos de gestión de refugios de animales**	Se analizarán diferentes enfoques y estrategias utilizadas en refugios para mejorar la eficiencia en la gestión.	Libros y artículos sobre administración de refugios de animales.
	2.4.2. Uso de sistemas de información en organizaciones sin fines de lucro	Se estudiará cómo los sistemas de información optimizan la gestión y mejoran la transparencia en ONGs y refugios.	Estudios sobre implementación de TIC en organizaciones sin fines de lucro.

FUENTE: Elaboración propia.

1.11.2 Índice Tentativo Comentado

ÍNDICE TENTATIVO

CAPÍTULO I - INTRODUCCIÓN

1.1 Introducción

1.2 Antecedentes

1.3 Planteamiento del Problema

1.3.1 Identificación de la Situación Problemática

1.4 Formulación del Problema

1.4.1 Análisis Causa – Efecto

1.5 Objetivos de la Investigación

1.5.1 Objetivo General

1.5.2 Objetivos Específicos

1.6 Justificación

1.6.1 Justificación Técnica

1.6.2 Justificación Económica

1.6.3 Justificación Social

1.7 Alcance

1.7.1 Alcance Temático

TABLA 8: CRONOGRAMA DE TRABAJO

Tarea	Descripción	Duración
1	Pulido del perfil	1 semana
2	Elaboración de los marcos del proyecto	1 semana
3	Realización de la entrevista	1 semana
4	Diseño de la base de datos	1 semana
5	Redacción final de los requerimientos	2 semanas
6	Elaboración del Manual de Funciones	3 semanas
7	Diseño del frontend	9 semanas
8	Diseño del backend	11 semanas
9	Subir al hosting	1 semana
10	Pruebas de seguridad	2 semanas

1.11.4 Plan de Trabajo para Elaboración del Trabajo de Grado

TABLA 9: PLAN DE TRABAJO PARA ELABORACIÓN DE TRABAJO DE GRADO

No.	Elementos por desarrollar	Actividades	Fecha de inicio	Duración (días)	Fecha de fin
1	Pulido del perfil	Revisión de observaciones recibidas del tribunal de defensa de perfil.	30/03/2025	1	01/04/2025
		Revisión de documentos complementarios	02/05/2025	3	05/04/2025
		Redacción del perfil pulido	06/10/2025	6	12/04/2025
2	Elaboración del Capítulo 1. Introducción.	Revisión de los principales componentes del capitulo	13/04/2025	3	16/04/2025
		Verificación de la redacción	17/04/2025	2	17/04/2025

No.	Elementos por desarrollar	Actividades	Fecha de inicio	Duración (días)	Fecha de fin
2	Elaboración del Capítulo 1. Introducción.	Establecer subtítulos en el proyecto	19/04/2025	2	20/04/2025
3	Elaboración del Capítulo 2. Marcos de la Investigación.	Revisión del marco teórico y conceptual	27/04/2025	5	26/04/2025
		Análisis de antecedentes y referencias	27/04/2025	5	01/05/2025
4	Elaboración del Capítulo 3.	Definición de tipo y alcance de la investigación	02/05/2025	4	05/05/2025
		Explicación de técnicas e instrumentos	06/05/2025	4	09/05/2025
5	Elaboración del Capítulo 4. Desarrollo del sistema	Diseño de base de datos y modelos	10/05/2025	7	16/05/2025
		Implementación del backend en Laravel	17/05/2025	20	05/06/2025
		Desarrollo del frontend en Vue.js	06/06/2025	20	25/06/2025
6	Elaboración del Capítulo 5. Pruebas y validación	Pruebas unitarias y de integración	26/06/2025	10	05/07/2025
7	Elaboración del Capítulo 6. Resultados y análisis	Evaluación del sistema implementado	06/07/2025	7	12/07/2025
8	Elaboración de conclusiones y recomendaciones	Redacción final del documento	13/07/2025	5	17/07/2025
9	Pulido general de todo el trabajo	Revisión ortográfica y de formato	18/07/2025	4	21/07/2025
10	Entrega final del trabajo	Presentación ante el tribunal	22/07/2025	1	22/07/2025

FUENTE: Elaboración propia

1.11.5 Diagrama de grantt

TABLA 10: DIAGRAMA DE GANTT DEL PROYECTO

Elementos	Febrero				Marzo				Abril				Mayo				Junio				Julio			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Actividad 1																								
Actividad 2																								
Actividad 3																								
Actividad 4																								
Actividad 5																								
Actividad 6																								
Actividad 7																								
Actividad 8																								
Actividad 9																								
Actividad 10																								

CAPÍTULO II

MARCO TEÓRICO

Marco teórico

2 Marco teórico

Un desafío constante para quienes investigan ha sido la necesidad de conocer a fondo los trabajos previos en su área de estudio. Esto no solo ayuda a ampliar el conocimiento y entender el contexto actual, sino que también evita la repetición de esfuerzos ya realizados. Por eso, es vital construir un marco teórico, que se define como la organización lógica y secuencial de información teórica proveniente de fuentes bibliográficas confiables, relacionadas con el problema de investigación, y que sirve como base para proponer soluciones (Garcia,2018).

2.2 Sistemas de Información

Los sistemas de información empresariales han evolucionado en los últimos años hasta convertirse en un pilar esencial dentro de las organizaciones. Ya no se limitan únicamente a ser un conjunto de software y hardware utilizados en la gestión operativa diaria, sino que su papel ha trascendido. Inicialmente vistos como herramientas para reducir la burocracia y agilizar las transacciones, ahora se consideran un recurso estratégico clave para alcanzar una ventaja competitiva sostenible. Por esta razón, es fundamental que las empresas los integren en su proceso de planificación estratégica, asegurando que su desarrollo y aplicación se alineen con sus necesidades informativas y estructura organizativa (Hernandez,2021).

2.3 Ingeniería de Software

La ingeniería de software es una rama de la ciencia de la información que emplea diversas metodologías de diseño, adaptándose a los requerimientos específicos de los usuarios o clientes. Cada sistema de información representa un diseño complejo, desarrollado para cumplir con una operatividad particular. Entre las metodologías ágiles destaca Scrum, caracterizado por su enfoque en la comunicación continua con el cliente, el desarrollo incremental, la flexibilidad del

proceso y la priorización de la funcionalidad del producto y su arquitectura sobre una documentación exhaustiva (Gordón,2022).

La ingeniería de software es fundamental en el desarrollo de sistemas de información, ya que permite aplicar metodologías que garantizan soluciones eficientes y adaptadas a las necesidades de los usuarios. Dado que cada sistema tiene características y requerimientos específicos, es necesario utilizar enfoques que permitan una adaptación flexible durante el proceso de desarrollo.

En este sentido, las metodologías ágiles, como Scrum, ofrecen una manera efectiva de gestionar proyectos al priorizar la comunicación continua con el cliente y la entrega gradual de funcionalidades. Esto facilita la mejora constante del producto y asegura que responda a las expectativas de los usuarios sin depender de una documentación excesiva.

La elección de un enfoque ágil es clave para lograr un desarrollo más dinámico y eficiente, ya que permite adaptarse a cambios de manera rápida y reducir el riesgo de errores o funcionalidades innecesarias. Por ello, resulta esencial considerar metodologías como Scrum en el desarrollo de sistemas de información, garantizando así su calidad y utilidad en contextos reales.

2.4 Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software son enfoques estructurados utilizados para planificar, diseñar, implementar y mantener sistemas de software de manera eficiente. Estas metodologías permiten optimizar los recursos, minimizar errores y mejorar la calidad del software entregado. A lo largo del tiempo, han surgido diferentes modelos de desarrollo, cada uno con sus propias ventajas y desventajas según el tipo de proyecto.

Entre las metodologías más utilizadas se encuentran el modelo en cascada y las metodologías ágiles como Scrum. El modelo en cascada sigue un enfoque lineal y

secuencial, donde cada fase del desarrollo debe completarse antes de pasar a la siguiente. Por otro lado, Scrum, como parte de las metodologías ágiles, se centra en la flexibilidad, la colaboración y la entrega incremental del producto, permitiendo adaptaciones constantes según los requerimientos del cliente.

La elección de una metodología depende de diversos factores como el tamaño del equipo, la complejidad del proyecto y la necesidad de iteraciones rápidas. Implementar una metodología adecuada no solo mejora la eficiencia del desarrollo, sino que también contribuye a la satisfacción del cliente y al éxito del proyecto.

El desarrollo de software es una tarea compleja que, durante mucho tiempo, se llevó a cabo sin seguir una metodología definida. Algunos autores describen una metodología como un conjunto de procesos, técnicas, herramientas y documentos de apoyo que facilitan a los desarrolladores la implementación de nuevos sistemas de información. En las últimas dos décadas, ha surgido un intenso debate en torno a las metodologías de desarrollo de software, dividiéndose en dos enfoques principales. Por un lado, las metodologías tradicionales, que se enfocan en el control del proceso y en un seguimiento meticuloso de cada etapa del desarrollo. Por otro lado, las metodologías ágiles, que priorizan la interacción humana, la colaboración con el cliente y la entrega continua de software mediante iteraciones cortas y frecuentes (Gomez,2014).

2.4.1 Modelo en Cascada

El modelo en cascada es un enfoque de desarrollo de software que sigue una secuencia estructurada, en la que cada fase del proceso se ejecuta de manera ordenada, una después de otra. Su nombre proviene de la forma en que las diferentes etapas del proyecto están organizadas, siguiendo un flujo descendente similar al de una cascada, este modelo tiene su origen en industrias como la construcción y la manufactura, donde realizar modificaciones después de finalizado un proceso es costoso y complejo. Dado que en sus inicios el desarrollo de software

no contaba con metodologías específicas, se adaptó este enfoque secuencial (Dominguez,2017).

El modelo en cascada fue ampliamente utilizado en el desarrollo de software en sus primeras etapas, debido a su estructura ordenada y fácil comprensión. Sin embargo, su rigidez y la dificultad para adaptarse a cambios lo han hecho menos viable en entornos donde la flexibilidad y la iteración son necesarias. Metodologías más modernas, como las ágiles, han ganado popularidad porque permiten una mayor colaboración con el cliente y una adaptación continua a los requerimientos cambiantes. Aun así, el modelo en cascada sigue siendo útil en proyectos donde los requisitos están bien definidos desde el inicio y es esencial seguir un proceso estructurado y documentado.

2.4.2 Metodología Ágil (Scrum)

Scrum es una metodología ágil utilizada en el desarrollo de software que se basa en la colaboración, la adaptabilidad y la entrega continua de valor. A diferencia de los enfoques tradicionales, Scrum organiza el trabajo en ciclos llamados sprints, donde se establecen objetivos concretos y funcionales en un período de tiempo corto, generalmente de dos a cuatro semanas.

Una de sus principales características es la comunicación constante entre los miembros del equipo y el cliente, lo que permite ajustar el desarrollo según las necesidades cambiantes del proyecto. Además, Scrum se apoya en reuniones diarias y eventos clave, como la planificación del sprint (Sprint Planning), la reunión diaria (Daily Scrum), la revisión del sprint (Sprint Review) y la retrospectiva (Sprint Retrospective), con el fin de optimizar el proceso y mejorar continuamente el producto final.

Scrum cuenta con roles bien definidos:

- **Product Owner**: Representa al cliente y se encarga de gestionar los requisitos del producto.
- **Scrum Master:** Facilita la metodología, eliminando obstáculos que puedan afectar al equipo y asegurando que se sigan los principios ágiles.

Equipo de Desarrollo: Son los encargados de diseñar, programar y probar el software en cada iteración.

Gracias a su enfoque iterativo e incremental, Scrum permite entregar versiones funcionales del software en cada sprint, asegurando una respuesta rápida a los cambios y una mejora continua del producto (Scrum.org,2024)

2.5 Bases de Datos

En la actualidad, las bases de datos relacionales se han convertido en una de las herramientas más utilizadas en la era de la información, permitiendo la gestión, manipulación y recuperación de datos en distintos ámbitos. Sin embargo, su evolución no fue inmediata, ya que pasaron varios años hasta alcanzar el nivel de desarrollo que poseen hoy en día. Inicialmente, el almacenamiento de datos se realizaba mediante tarjetas perforadas y cintas magnéticas, luego se implementaron discos que marcaron el inicio de las bases de datos, seguidas por los modelos de bases de datos en red y jerárquicas. Posteriormente, surgió el modelo relacional, concebido inicialmente como un conjunto de principios para evaluar los administradores de sistemas de datos basados en relaciones. En sus inicios, este modelo no tuvo gran aceptación debido a que su rendimiento era inferior en comparación con otros sistemas de almacenamiento. Esto se debía a que utilizaba tablas para organizar la información, dejando de lado elementos clave como las claves primarias, lo que limitaba su eficacia y desempeño en sus primeras versiones (Córdoba,2023).

2.5.1 Bases de Datos Relacionales

Este modelo se utiliza para representar problemas reales y gestionar datos de manera flexible. Su principio central es la interconexión de las entidades, representadas como tablas, que contienen registros llamados tuplas. Cada tabla tiene filas que corresponden a estas tuplas, y cada tupla está formada por campos, que son las columnas de la tabla. La estructura de estas relaciones facilita la visualización de los datos y su organización, permitiendo una gestión eficiente y comprensible (Córdoba,2023).

El modelo relacional de bases de datos es un enfoque ampliamente utilizado por su simplicidad y efectividad. Al estructurar los datos en tablas con relaciones entre ellas, se facilita el acceso, actualización y mantenimiento de la información. La idea de representar los datos de manera intuitiva, como tuplas dentro de tablas, hace que este modelo sea comprensible y práctico para una gran variedad de aplicaciones. Además, la capacidad de crear relaciones entre las tablas mediante claves primarias y foráneas permite una organización más clara y eficiente, optimizando la gestión de datos en sistemas complejos.

Algunos de los gestores de bases de datos relacionales más utilizados en la industria son:

MySQL: Un sistema de gestión de bases de datos de código abierto muy popular, utilizado en aplicaciones web.

PostgreSQL: Un RDBMS de código abierto conocido por su robustez y conformidad con los estándares SQL.

Oracle Database: Un RDBMS comercial de alto rendimiento, utilizado en grandes empresas con necesidades complejas.

Microsoft SQL Server: Un sistema de bases de datos de Microsoft utilizado principalmente en entornos empresariales y aplicaciones críticas.

2.5.2 Bases de Datos No Relacionales

Las bases de datos no relacionales (NoSQL) son un tipo de sistema de gestión de bases de datos que se aleja del modelo tabular utilizado por las bases de datos relacionales. Estas bases de datos están diseñadas para manejar grandes volúmenes de datos que pueden no encajar bien en el modelo relacional tradicional, permitiendo una mayor flexibilidad y escalabilidad.

Existen varios tipos de bases de datos NoSQL, cada uno con un modelo de datos distinto, adecuado para diferentes tipos de aplicaciones:

Bases de Datos de Clave-Valor: Almacenan datos como un conjunto de pares clave-valor, donde cada clave es única y se asocia con un valor específico. Son ideales para almacenamiento rápido y consultas simples.

Ejemplo: Redis, Amazon DynamoDB.

Bases de Datos de Documento: Almacenan datos en documentos (normalmente en formato JSON o BSON), lo que permite una estructura flexible y más compleja que las bases clave-valor.

Ejemplo: MongoDB, CouchDB.

Bases de Datos de Columnas: Estas bases de datos son los que organizan los datos en columnas en lugar de filas, lo que es más eficiente para consultas que involucran grandes volúmenes de datos y requieren lecturas rápidas de columnas específicas.

Ejemplo: Apache Cassandra, HBase.

Bases de Datos de Grafos: Estas bases de datos están diseñadas para almacenar y procesar relaciones complejas entre datos, representando los datos como nodos, aristas y propiedades.

Ejemplo: Neo4j, ArangoDB.

1.1.2. Selección del Gestor de Base de Datos (MySQL)

TABLA N°11: Bases de datos.

Característica	MySQL	PostgreSQL	SQL Server	MongoDB (NoSQL)	SQLite
Modelo de datos	Relacional (SQL)	Relacional (SQL)	Relacional (SQL)	No relacional (Documentos JSON)	Relacional (SQL)
Rendimiento	Rápido en consultas comunes	Más potente en operaciones complejas	Bueno para entornos empresariales	Excelente en grandes volúmenes de datos	Ligero y rápido para aplicaciones pequeñas
Escalabilidad	Vertical (mejorando hardware)	Vertical y horizontal	Vertical (requiere hardware potente)	Horizontal (distribuido)	Limitada
Facilidad de uso	Fácil de aprender y administrar	Más complejo, pero potente	Requiere configuración avanzada	Requiere conocimientos en NoSQL	Muy fácil, sin configuración
Soporte de transacciones (ACID)	Sí	Sí (más avanzado que MySQL)	Sí	No en todas las operaciones	Sí
Integración con Laravel	Excelente	Buena (requiere más configuración)	Compatible, pero menos común	Necesita adaptaciones	Soportado, pero no recomendado
Costo	Gratuito y de código abierto	Gratuito y de código abierto	Pago (licencia de Microsoft)	Gratuito y de código abierto	Gratuito y de código abierto
Casos de uso	Aplicaciones web, bases de datos medianas	Aplicaciones empresariales, análisis de datos	Aplicaciones empresariales grandes	Big Data, sistemas en la nube, redes sociales	Aplicaciones móviles y software ligero

FUENTE: Elaboración propia

Después de analizar diversas opciones, se ha determinado que MySQL es la mejor elección, ya que es una base de datos relacional de código abierto que proporciona un equilibrio adecuado entre rendimiento, facilidad de uso y compatibilidad con Laravel, el framework utilizado para el desarrollo del sistema.

2.6 Paradigmas de Programación

Un paradigma de programación define un enfoque específico para resolver problemas computacionales, estableciendo las reglas sobre cómo deben organizarse y estructurarse las tareas dentro de un programa (Rodriguez,2011).

Los paradigmas de programación son fundamentales porque determinan la forma en que diseñamos y desarrollamos software. Dependiendo del problema a resolver, podemos elegir entre distintos paradigmas como el estructurado, orientado a objetos o funcional. En mi proyecto, utilizo un enfoque basado en el paradigma orientado a objetos porque facilita el modularidad y el mantenimiento del código.

2.6.1 Programación Orientada a Objetos

Desde la década de 1970, surgieron nuevos lenguajes de simulación y herramientas para la construcción de prototipos, como Simula-70 y Smalltalk, este último influenciado por el primero. En estos lenguajes, la abstracción de datos juega un papel clave, permitiendo representar problemas del mundo real a través de objetos que combinan datos y las operaciones que pueden realizar. Conceptos fundamentales como la abstracción de datos, los objetos y la encapsulación forman la base de la Programación Orientada a Objetos (Gonzáles, 2004).

La evolución de los lenguajes de programación ha buscado siempre una mayor eficiencia en la representación de problemas. La Programación Orientada a Objetos (POO) nos permite crear modelos de software que son más fáciles de entender y mantener, gracias a la encapsulación de datos y comportamiento en objetos. En mi caso, este paradigma fue elegido por su capacidad para facilitar la reutilización de

código y mejorar la organización del software, lo que resulta en aplicaciones más escalables.

2.6.2 Programación Funcional

La Programación Funcional (PF) es un paradigma de programación que se basa en el uso de funciones matemáticas para construir programas. A diferencia de otros paradigmas como la Programación Orientada a Objetos (POO), la PF se centra en "qué" se debe hacer, en lugar de "cómo" se debe hacer.

2.6.3 Programación lógica

La Programación Lógica es un paradigma de programación que se basa en la lógica formal para representar y resolver problemas. A diferencia de otros paradigmas como la programación imperativa (que se centra en "cómo" hacer algo) o la programación funcional (que se centra en "qué" hacer), la programación lógica se centra en "qué es verdadero".

En el ámbito de la programación lógica, los elementos fundamentales que se pueden declarar son hechos y reglas. Un hecho representa una conexión específica entre objetos individuales, mientras que una regla establece una relación general aplicable a objetos que comparten ciertas características. Para expresar una relación entre objetos, se utiliza el nombre de la relación seguido de los objetos involucrados, encerrados entre paréntesis. Por ejemplo: Hijo (Juan, Luis) (Cerrada & Collado, 2005).

2.7 Lenguajes y Frameworks de Desarrollo

La elección del lenguaje de programación y los frameworks es un factor crítico en el desarrollo de software, ya que determina la eficiencia, la capacidad de expansión y la facilidad de mantenimiento del sistema. Hay una amplia gama de lenguajes, cada uno diseñado con características únicas que se adaptan mejor a ciertos tipos de proyectos.

Para la parte del servidor de este sistema, se decidió utilizar PHP con el framework Laravel. Esto se debe a que Laravel es fácil de usar, tiene una gran comunidad de desarrolladores que lo respaldan y permite crear aplicaciones con una estructura modular y segura. Laravel ofrece herramientas que facilitan tareas como el manejo de rutas, la autenticación de usuarios, la interacción con bases de datos y la seguridad, lo que acelera el desarrollo y reduce la necesidad de escribir mucho código repetitivo.

En cuanto a la interfaz de usuario, se optó por Vue.js, un framework de JavaScript que permite crear interfaces dinámicas y que reaccionan a los cambios. Vue.js es ligero, se integra fácilmente con otros proyectos y ofrece un buen rendimiento, lo que lo hace ideal para mejorar la experiencia del usuario en la aplicación.

2.7.1 Backend: PHP y Laravel

Existen numerosas herramientas y frameworks para construir aplicaciones web, pero Laravel destaca como una opción sobresaliente para proyectos modernos y completos (Laravel.com, 2025).

Laravel se adapta a diferentes niveles de experiencia. Para quienes se inician en el desarrollo web, su amplia documentación y tutoriales facilitan el aprendizaje. Los desarrolladores experimentados encuentran en Laravel herramientas avanzadas para tareas complejas, como la inyección de dependencias y las pruebas unitarias (Laravel.com, 2025).

Laravel también es altamente escalable, gracias a su compatibilidad con sistemas de caché distribuidos como Redis. Esto permite que las aplicaciones Laravel manejen grandes volúmenes de tráfico. Para proyectos que requieren una escalabilidad extrema, existen plataformas como Laravel Cloud (Laravel.com, 2025).

Además, Laravel cuenta con una gran comunidad de desarrolladores que contribuyen activamente al framework, lo que garantiza su constante evolución y mejora (Laravel.com, 2025).

2.7.2 Frontend: Vue.js

Con el crecimiento exponencial de la web y la diversidad de lenguajes de programación, la elección de la tecnología adecuada se ha vuelto crucial para cada proyecto. Este artículo se centra en Vue.js, un framework progresivo para el desarrollo frontend, y analiza sus ventajas en comparación con otros frameworks de JavaScript populares entre la comunidad y las grandes empresas.

Vue.js se presenta como una opción versátil y escalable, adaptándose a las necesidades de proyectos de diferentes tamaños y complejidades. Su diseño progresivo permite una integración gradual, facilitando el aprendizaje para principiantes y ofreciendo herramientas avanzadas para desarrolladores experimentados.

Además, Vue.js destaca por su rendimiento y su capacidad para crear interfaces de usuario dinámicas y reactivas, mejorando la experiencia del usuario. Su ecosistema rico y su comunidad activa lo convierten en una opción atractiva para el desarrollo frontend moderno (Garcia, 2019).

En el contexto del desarrollo de aplicaciones y páginas web, la proliferación de tecnologías frontend demanda una evaluación rigurosa de las herramientas disponibles. Este documento analiza Vue.js, un framework progresivo de JavaScript, y su idoneidad para la construcción de interfaces de usuario modernas.

Vue.js se distingue por su arquitectura escalable y su capacidad de adaptación a proyectos de diversa complejidad. Su diseño progresivo facilita la incorporación gradual de funcionalidades, permitiendo tanto a desarrolladores noveles como experimentados aprovechar sus capacidades.

Además, Vue.js ofrece un rendimiento optimizado para la creación de interfaces dinámicas y reactivas, contribuyendo a una experiencia de usuario mejorada. Su ecosistema robusto y su comunidad activa consolidan su posición como una herramienta relevante en el desarrollo frontend contemporáneo.

2.7.3 Frameworks de Diseño

2.7.3.1 Bootstrap

Bootstrap es un framework de código abierto para el desarrollo frontend de sitios y aplicaciones web. En términos sencillos, es un conjunto de herramientas y estilos predefinidos que facilitan la creación de interfaces de usuario atractivas y responsivas.

2.7.3.2 Tailwind CSS

Tailwind CSS es un framework de CSS utilitario que se diferencia de otros frameworks como Bootstrap. En lugar de proporcionar componentes predefinidos (como botones o barras de navegación), Tailwind CSS ofrece una colección de clases de utilidad de bajo nivel que se puede combinar para construir diseños personalizados directamente en HTML.

2.8 Seguridad en Aplicaciones Web

En la era digital, la protección de la información se ha vuelto indispensable para las empresas, dado que los datos corporativos se procesan, transfieren y almacenan principalmente en aplicaciones y servicios web. Por lo tanto, es imperativo implementar metodologías y procedimientos que salvaguarden la información de posibles ataques a estos sistemas. Este artículo examina las principales debilidades de seguridad presentes en las aplicaciones web y propone estrategias para mitigar estos riesgos, buscando fortalecer la seguridad de los sistemas web utilizados por organizaciones y empresas a nivel global (Zambrano, 2019).

La seguridad en aplicaciones web es un aspecto fundamental en el desarrollo de software, ya que cualquier vulnerabilidad puede comprometer la integridad de los datos y la privacidad de los usuarios. Entre los principales riesgos de seguridad se encuentran:

Autenticación y Autorización: Implementación de mecanismos como JWT (JSON Web Token) o OAuth para garantizar el acceso seguro a los recursos.

Cifrado de Datos: Uso de SSL/TLS para la protección de la información transmitida entre el servidor y los clientes.

Prevención de Inyección SQL: Uso de consultas preparadas y ORM para evitar la manipulación maliciosa de las bases de datos.

Protección contra XSS (Cross-Site Scripting): Implementación de medidas para evitar la inserción de scripts maliciosos en las páginas web.

Protección contra CSRF (Cross-Site Request Forgery): Uso de tokens CSRF para evitar solicitudes malintencionadas en nombre del usuario.

2.9 Arquitectura de Software

La Arquitectura de Software (AS) es una disciplina fundamental en el desarrollo de sistemas, ya que establece una visión estructurada sobre la organización y diseño de aplicaciones. A pesar de su importancia, la literatura en español sobre el tema es aún limitada y suele quedar desactualizada rápidamente debido a la constante evolución tecnológica.

La arquitectura de software define la estructura y diseño de un sistema, facilitando su mantenimiento, escalabilidad y reutilización de código. Existen diversas arquitecturas, entre ellas:

Arquitectura Monolítica: Todo el sistema está contenido en una única aplicación.

Arquitectura por Capas: División del sistema en capas como presentación, lógica de negocio y datos.

Arquitectura MVC (Modelo-Vista-Controlador): Organiza el código en tres componentes separados, facilitando la escalabilidad y el mantenimiento.

Arquitectura Basada en Microservicios: Separa los módulos de la aplicación en pequeños servicios independientes.

Para este proyecto se ha elegido MVC debido a que proporciona una mejor separación de responsabilidades, lo que facilita el mantenimiento del código y la incorporación de nuevas funcionalidades sin afectar la estructura general.

2.10 Servicios en la Nube y Hosting

El despliegue de una aplicación requiere la elección de un entorno de alojamiento adecuado. Algunas de las opciones más utilizadas son:

cPanel: Un panel de control de hosting tradicional que facilita la gestión de sitios web.

AWS (Amazon Web Services): Ofrece servidores escalables y seguros para el despliegue de aplicaciones.

DigitalOcean: Proporciona máquinas virtuales (droplets) con alta disponibilidad y control total sobre el servidor.

VPS (Servidor Privado Virtual): Ofrece más flexibilidad y control en comparación con un hosting compartido.

Para este proyecto, se considerará una opción de hosting que brinde estabilidad, escalabilidad y seguridad.

2.11 APIs y Consumo de Servicios Web

Las APIs (Application Programming Interfaces) permiten la comunicación entre diferentes aplicaciones o servicios. Existen dos tipos principales:

RESTful APIs: Basadas en HTTP, permiten operaciones como GET, POST, PUT y DELETE.

GraphQL: Permite consultas más eficientes al proporcionar solo los datos necesarios.

Para este proyecto, se implementará una API RESTful con Laravel, lo que permitirá la integración con otros sistemas y facilitará el consumo de datos en la aplicación frontend desarrollada en Vue.js.

2.12 Inteligencia Artificial en Chatbots

El uso de chatbots en aplicaciones web ha crecido significativamente debido a su capacidad de mejorar la experiencia del usuario mediante respuestas automáticas y asistencia 24/7. Los chatbots pueden implementarse de varias formas:

Chatbots basados en reglas: Responden según palabras clave predefinidas.

Chatbots con procesamiento de lenguaje natural (NLP): Utilizan IA para entender y generar respuestas más naturales.

Integración con plataformas como Dialogflow o ChatGPT: Permiten la implementación de chatbots avanzados con aprendizaje automático.

En este proyecto, se busca implementar un chatbot inteligente para asistir a los adoptantes y responder preguntas sobre el refugio y los procesos de adopción.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Cerrada, J., & Collado, M. (2005). Fundamentos de programación. Editorial Universitaria Ramón Areces.

González, Abdiel E Cáceres. (2004). Centro de Investigación y de Estudios Avanzados

García, Raúl. (2019). Estudio de la popularidad del framework VueJS

Córdova, Fernanda. (2013). Análisis comparativo entre bases de datos relacionales con bases de datos no relacionales

Dominguez, Pablo. (2017). En qué consiste el modelo en cascada - Gestiona tu proyecto de desarrollo – OpenClassrooms

Tinoco Gómez, Oscar. (2014). Criterios de selección de metodologías de desarrollo de software

Rivera García, Patricia. (2018). Marco Teórico, Elemento Fundamental en el Proceso de Investigación Científica

Zambrano, Alex & Guarda, Teresa (2019). Técnicas de mitigación para principales vulnerabilidades de seguridad en aplicaciones web

Cerrada, J., & Collado, M. (2005). Fundamentos de programación. Editorial Universitaria Ramón Areces.

González, Abdiel E Cáceres. (2004). Centro de Investigación y de Estudios Avanzados

García, Raúl. (2019). Estudio de la popularidad del framework VueJS

Córdova, Fernanda. (2013). Análisis comparativo entre bases de datos relacionales con bases de datos no relacionales

Dominguez, Pablo. (2017). En qué consiste el modelo en cascada - Gestiona tu proyecto de desarrollo – OpenClassrooms

Tinoco Gómez, Oscar. (2014). Criterios de selección de metodologías de desarrollo de software

Rivera García, Patricia. (2018). Marco Teórico, Elemento Fundamental en el Proceso de Investigación Científica

Zambrano, Alex & Guarda, Teresa (2019). Técnicas de mitigación para principales vulnerabilidades de seguridad en aplicaciones web

WEBLOGRAFÍA

https://www.fundacion-affinity.org/es/principales-razones-de-abandono-de-un-animal-de-compania

ANEXOS