Trabajo Práctico N°2

Nombre y apellido: Florencia Otero

Legajo: VINF015907

Cátedra: CAT C - INF275 - EDH

Materia: Seminario de práctica de informática

Profesor experto y disciplinar: Hugo Fernando Frias y Pablo Alejandro Virgolini

Fecha de entrega: 05/10/2025

Índice

PRIMERA ENTREGA

SEGUNDA ENTREGA:

Objetivos

- Definir el alcance y justificación de un proyecto informático para dar solución a una situación problemática.
- Realizar la justificación de un proyecto y la definición de objetivos.
- Aplicar el proceso unificado de desarrollo (PUD).
- Realizar el análisis del modelo de negocio.
- Plantear requerimientos funcionales y no funcionales.

Enunciado

Los sistemas informáticos desempeñan un papel muy importante para la optimización de procesos en diversas áreas de cualquier organización. La tecnología brinda la posibilidad de automatizar tareas, recopilar y analizar datos a gran escala, agilizar procesos y proporcionar soluciones a desafíos complejos.
Te propongo que determines con claridad un problema que pueda resolverse con la implementación de un proyecto informático y realices una entrega de acuerdo con lo solicitado en la consigna. Puedes definir cualquier organización real y explorar oportunidades en relación con su seguridad, logística, gestión de inventarios, optimización de procesos industriales, análisis de datos, cuidado de la salud, la educación, o cualquier otra área.
Para la realización de un proyecto de desarrollo informático, se requiere realizar un profundo análisis del negocio y comprender la dinámica de la organización. De esta manera, será posible comprender correctamente las necesidades, identificar los procesos, flujos de trabajo y desafíos que la organización enfrenta a diario y que pueden optimizarse con la aplicación de un desarrollo. Servirá también como base para justificar el proyecto, definir el alcance y objetivos.
Es clave llevar adelante un correcto proceso de elicitación, que involucra la definición de requerimientos funcionales y no funcionales.
Una vez completada esta fase, se avanza en el análisis y diseño detallado del sistema. Esto implica la creación de modelos, la definición de la arquitectura, la planificación de la estructura de datos y la lógica de funcionamiento. Solo con un correcto análisis y diseño se puede garantizar que el sistema se construirá de manera eficiente y que cumplirá con los objetivos establecidos.

Consignas

- Título del proyecto.
- Introducción.
- Justificación.
- Definiciones del proyecto y del sistema.
- Elicitación.
- Conocimiento del negocio.
- Propuesta de solución.
- Inicio del análisis: Requerimientos funcionales, no funcionales y casos de uso.

Título del proyecto

Desarrollo de un sistema informático para la gestión y difusión de mascotas perdidas y animales callejeros (perros y gatos) en la ciudad de Córdoba Capital.

Introducción

La problemática de los animales callejeros y las mascotas perdidas constituye una realidad creciente en la ciudad de Córdoba. Miles de perros y gatos viven en las calles expuestos a enfermedades, accidentes, maltrato y condiciones precarias, mientras que muchas familias sufren la pérdida de sus animales de compañía sin contar con herramientas eficaces para encontrarlos.

En este contexto, las organizaciones de rescate y los refugios enfrentan grandes limitaciones, ya que dependen de donaciones, voluntariado y difusión en redes sociales, las cuales no siempre resultan suficientes ni organizadas.

Con el fin de aportar una solución tecnológica a esta situación, se plantea el desarrollo de un sistema de gestión que permita registrar, difundir y gestionar casos de mascotas perdidas y animales callejeros. Además, la plataforma incorporará un sistema de micro aportes económicos voluntarios que se destinarán a la atención veterinaria, castraciones y mantenimiento de refugios.

Este proyecto se encuadra en la búsqueda de optimizar procesos de comunicación, gestión y colaboración comunitaria mediante una solución informática basada en buenas prácticas de desarrollo y con potencial de escalabilidad.

Cabe destacar que, para esta entrega, se han considerado y aplicado las correcciones señaladas por el profesor en el trabajo anterior. A partir de dichas mejoras, se incorporan en este documento los nuevos entregables correspondientes al Trabajo Práctico 2, en el marco de una propuesta incremental que amplía y enriquece el desarrollo iniciado previamente.

Justificación

La situación de las mascotas perdidas y los animales callejeros en Córdoba representa una preocupación creciente tanto para familias como para organizaciones de rescate. La ausencia de una herramienta tecnológica específica dificulta la comunicación eficiente entre quienes pierden o encuentran animales, y limita la capacidad de los refugios para recibir aportes que les permitan sostener su labor.

La solución propuesta busca responder a esta necesidad, aportando beneficios:

- **Sociales y comunitarios:** fortaleciendo la colaboración entre ciudadanos.
- **Sanitarios:** reduciendo la cantidad de animales en situación de calle.
- **Económicos:** canalizando micro aportes voluntarios para apoyar a los refugios.
- **Tecnológicos:** incorporando herramientas digitales accesibles a la comunidad.

De esta forma, el proyecto se justifica en su capacidad de generar un impacto positivo en la calidad de vida de las mascotas, de los vecinos y de las organizaciones involucradas.

Definiciones del proyecto y del sistema

Definiciones del proyecto

Objetivo general del proyecto

Proveer una solución informática que facilite la gestión y difusión de casos de mascotas perdidas y animales callejeros en Córdoba, promoviendo la colaboración ciudadana y el apoyo económico a través de micro donaciones voluntarias.

Objetivos específicos del proyecto

- Diseñar y documentar los requerimientos del sistema (funcionales y no funcionales), junto con casos de uso que permitan cubrir el registro, difusión y gestión de mascotas perdidas y animales callejeros.
- Implementar un prototipo funcional en Java con base de datos MySQL, que incluya los módulos de reportes, donaciones y gestión de usuarios/administradores.
- Evaluar y validar el prototipo mediante pruebas con usuarios reales del refugio y comunidad, recolectando retroalimentación para mejorar la usabilidad y la eficacia del sistema.

Explicación del cronograma y el diagrama de Gantt

El cronograma del proyecto se representa mediante un diagrama de Gantt, que es una herramienta de gestión utilizada para planificar y visualizar las actividades en el tiempo. Consiste en un gráfico de barras donde cada actividad se ubica en un período determinado, lo que permite observar con claridad las fechas de inicio, finalización y la duración de cada tarea.

En este caso, el diagrama refleja las cuatro entregas parciales definidas por la cátedra. Sin embargo, es importante destacar que, en la práctica, algunas actividades pueden superponerse. Por ejemplo, en la semana en que se trabaja en la primera entrega, también se puede comenzar a avanzar en tareas de la segunda, mientras se terminan de ajustar detalles de la primera.

Este solapamiento es propio de la dinámica real de un proyecto, donde los tiempos de desarrollo no siempre son lineales. De esta manera, aunque el cronograma establece fechas concretas de entrega, el diagrama de Gantt incorpora cierta flexibilidad, permitiendo organizar mejor el trabajo y asegurar la calidad de cada etapa.

	Agosto				Septiembre				Octubre					Noviembre			
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 1	Semana 2	Semana 3	Semana 4
Entrega 1: Definición Inicial - Justificación - Requerimientos - Negocio - Casos de uso																	
Entrega 2: Análisis - Diseño - Implementación - Pruebas - Base de datos - Comunicación																	
Entrega 3: Desarrollo - Java - Ejecución																	
Entrega 4: Proyecto final - Presentación - Video																	

Definiciones del sistema

Objetivo general del sistema

Desarrollar un sistema que permita registrar, organizar y difundir reportes de mascotas perdidas y animales callejeros en Córdoba, integrando funcionalidades de gestión de usuarios, publicación en redes sociales y administración de micro donaciones para el sostenimiento de los refugios.

Límites, alcances y restricciones del sistema

Límites:

- Desde: el registro y difusión de casos de mascotas perdidas o encontradas en la ciudad de Córdoba.
- Hasta: la gestión de donaciones internas al refugio y la publicación de reportes en redes sociales vinculadas.

Alcances:

- Usuarios podrán registrar y consultar reportes de mascotas perdidas o encontradas.
- El refugio tendrá acceso a una gestión centralizada de los casos y a los aportes recibidos.
- El sistema permitirá la difusión organizada en la plataforma y hacia redes sociales.
- Se gestionarán donaciones comunitarias destinadas al refugio y a la atención de animales callejeros.
- Los administradores podrán supervisar, validar y mantener la información registrada.

Restricciones:

- Funcionamiento sujeto a la disponibilidad de conexión a internet.
- Los usuarios deberán registrarse para acceder a todas las funciones.
- El sistema deberá garantizar la protección de datos sensibles y la transparencia en el manejo de aportes.
- Recursos limitados del refugio condicionan la capacidad operativa del sistema (voluntariado, fondos, equipamiento).

Elicitación

La elicitación es el proceso de adquirir “todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema” (Loucopoulos, 1995, como se citó en Oliveros y Antonelli, 2015, p. 2). Para llevarla adelante, se definió un conjunto de actividades que permitieron comprender la problemática de las mascotas perdidas, los animales callejeros y la gestión de recursos en un pequeño refugio.

Técnicas utilizadas

Encuestas:

Se diseñó un cuestionario en línea distribuido a través de redes sociales y grupos comunitarios de Córdoba. Su objetivo fue relevar información cuantitativa acerca de:

- Frecuencia con la que los usuarios encuentran animales callejeros.
- Experiencias previas en la pérdida de mascotas.
- Nivel de confianza en redes sociales como canal de búsqueda.
- Predisposición a realizar micro aportes económicos para colaborar con refugios.

Esto permitió identificar patrones generales y necesidades recurrentes de la comunidad.

Entrevistas:

Se realizaron entrevistas semiestructuradas con tres grupos de interés:

- **Dueños de mascotas:** para conocer las principales dificultades al momento de reportar o buscar un animal perdido.
- **Voluntarios y rescatistas:** para comprender cómo se organizan actualmente en la atención de animales callejeros.
- **Responsables del refugio:** para identificar los procesos internos de gestión (registro de ingresos, gastos veterinarios, castraciones, adopciones).

Estas entrevistas aportaron información cualitativa en profundidad sobre expectativas, limitaciones y posibles mejoras que un sistema informático podría brindar.

Herramientas empleadas

- Formularios de Google Forms para las encuestas.
- Grabaciones y transcripciones de entrevistas mediante Google Meet y WhatsApp.
- Análisis de resultados con hojas de cálculo (Excel/Google Sheets) para tabular respuestas y generar gráficos.

Participantes

- Usuarios finales (dueños de mascotas y comunidad).
- Voluntarios y rescatistas independientes.
- Administradores del refugio.
- Equipo de desarrollo encargado de sistematizar la información.

Análisis de competencia

Se analizaron plataformas existentes como Patitas Perdidas, grupos de Facebook e Instagram dedicados a animales extraviados y aplicaciones internacionales de adopción.

- Fortalezas: difusión masiva y rápida.
- Debilidades: falta de organización de la información, ausencia de trazabilidad de casos, escasa integración con refugios locales.

Esto evidenció la oportunidad de un sistema local más organizado y sostenible.

Tecnologías TIC consideradas

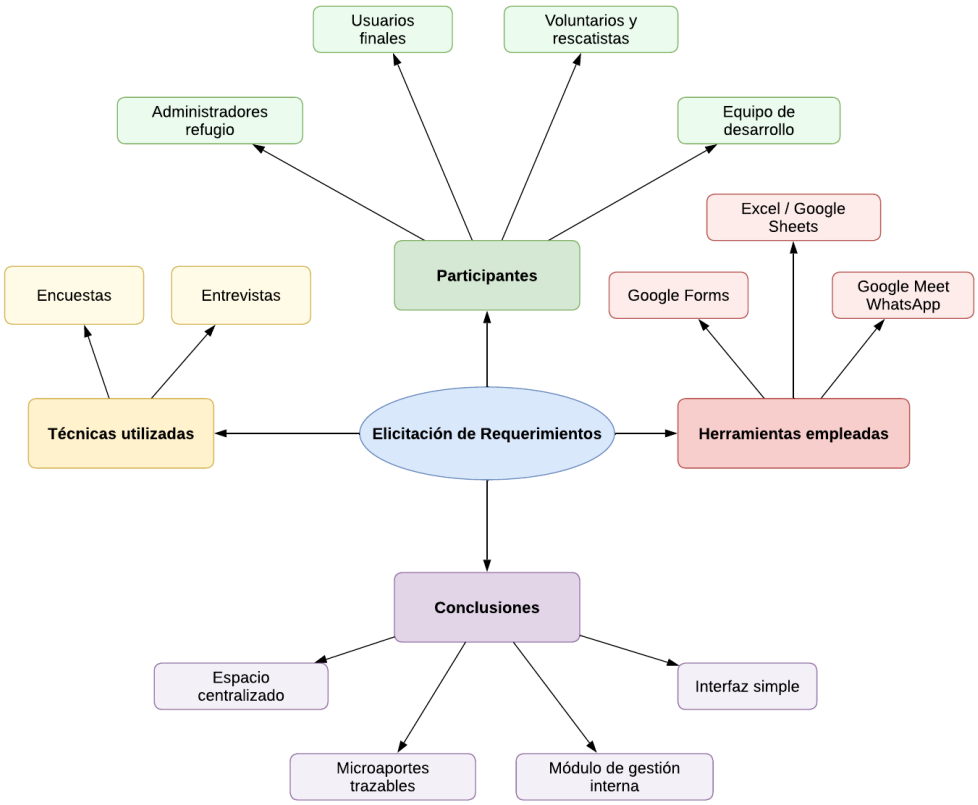
- Base de datos en MySQL para el almacenamiento persistente.
- Integración con APIs de pago electrónico para gestionar aportes.
- Difusión automatizada en redes sociales (Facebook, Instagram) mediante conectores.
- Java para el desarrollo del sistema.
- Swing/JavaFX o consola para la interfaz básica del prototipo.
- JDBC para la conexión entre la aplicación y la base de datos.
- GitHub como repositorio para el control de versiones y entrega del prototipo.

Conclusiones del relevamiento

Del proceso de elicitación se concluyó que:

- Existe una fuerte necesidad de un espacio centralizado para reportar y consultar casos de mascotas perdidas y animales callejeros.
- Los usuarios demandan una interfaz simple y accesible, dado que muchos no poseen conocimientos técnicos avanzados.
- El refugio requiere un módulo de gestión interna para registrar animales ingresados, donaciones recibidas y gastos médicos.
- La comunidad mostró gran interés en contar con un sistema de micro aportes transparentes y trazables.

A continuación, se presenta un diagrama que sintetiza los resultados del proceso de elicitación.



Conocimiento del negocio

La problemática de los animales callejeros y las mascotas perdidas se aborda actualmente de manera descentralizada a través de publicaciones en redes sociales, grupos comunitarios y la acción voluntaria de rescatistas y refugios.

El refugio participante en este proyecto cumple un rol fundamental, ya que recibe animales en situación de calle, les brinda atención veterinaria básica y busca hogares adoptivos. Sin embargo, la gestión de ingresos, egresos, donaciones y casos de adopción se realiza de forma manual y poco organizada.

Por otro lado, los dueños de mascotas que pierden a sus animales dependen de la difusión en redes sociales, lo que muchas veces resulta ineficiente debido al gran volumen de publicaciones y la falta de trazabilidad de los casos.

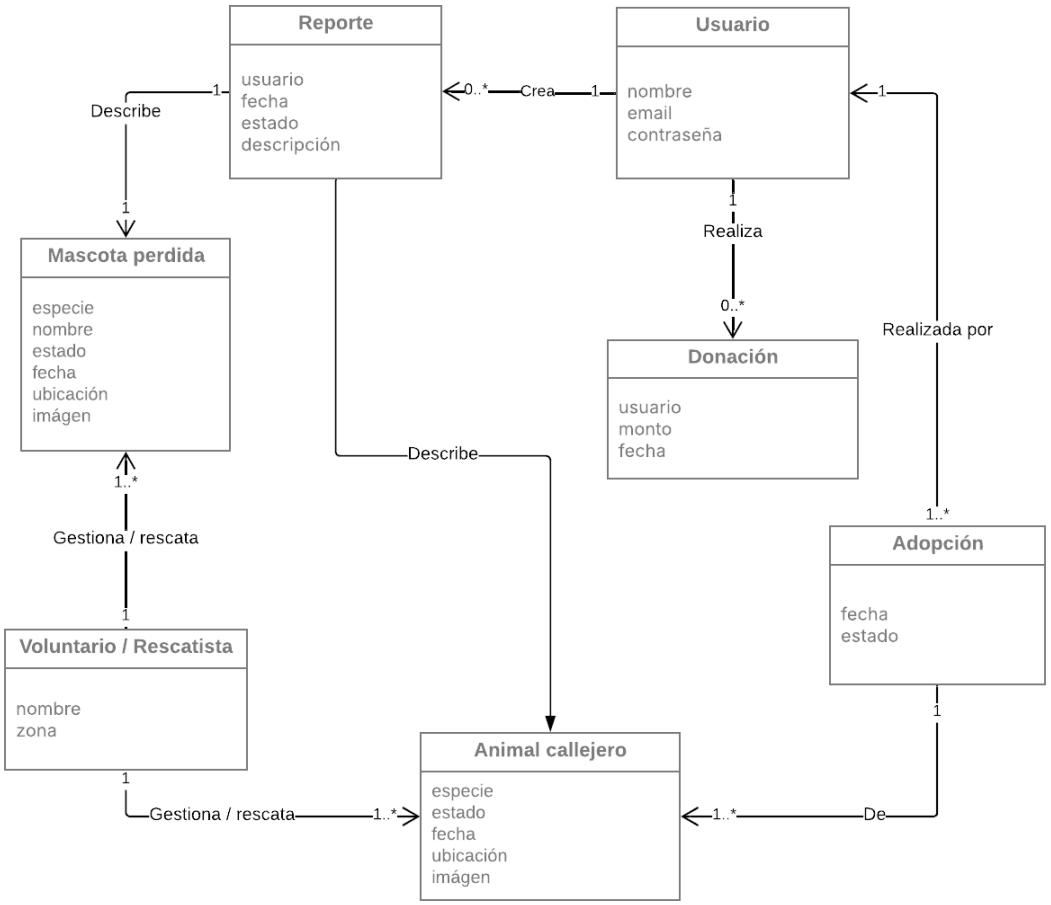
Actualmente, el “negocio” o dominio del problema se caracteriza por:

- **Procesos informales:** falta de un registro centralizado de animales perdidos o encontrados.
- **Dependencia de redes sociales:** la difusión se basa en publicaciones que se pierden con el tiempo.
- **Escasez de recursos en refugios:** los aportes económicos dependen de campañas puntuales y no de un sistema de micro donaciones recurrentes.
- **Alta participación comunitaria:** gran predisposición de la sociedad para ayudar, pero sin herramientas tecnológicas que canalicen el esfuerzo.

El valor del sistema a desarrollar radica en profesionalizar y organizar estos procesos, ofreciendo:

- Una plataforma centralizada para registrar, buscar y difundir casos.
- Un módulo de gestión interna para el refugio.
- Un sistema transparente de micro donaciones, con reportes que generen confianza en la comunidad.
- Una mayor trazabilidad y rapidez en la resolución de casos de mascotas perdidas y rescates.

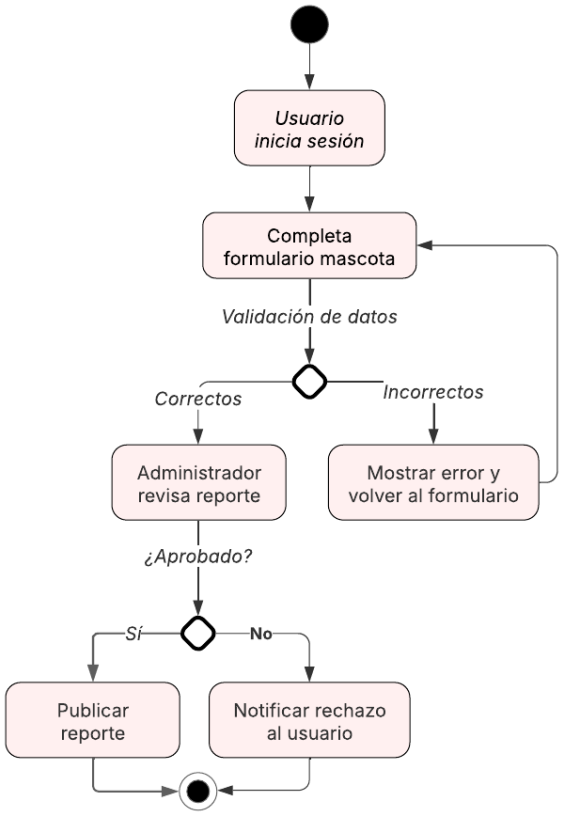
A continuación, se presenta el diagrama de dominio, representa las entidades principales y sus relaciones en el sistema de gestión de mascotas perdidas y animales callejeros. Está orientado al modelo de negocio, no al código del software.



Relevamiento de procesos

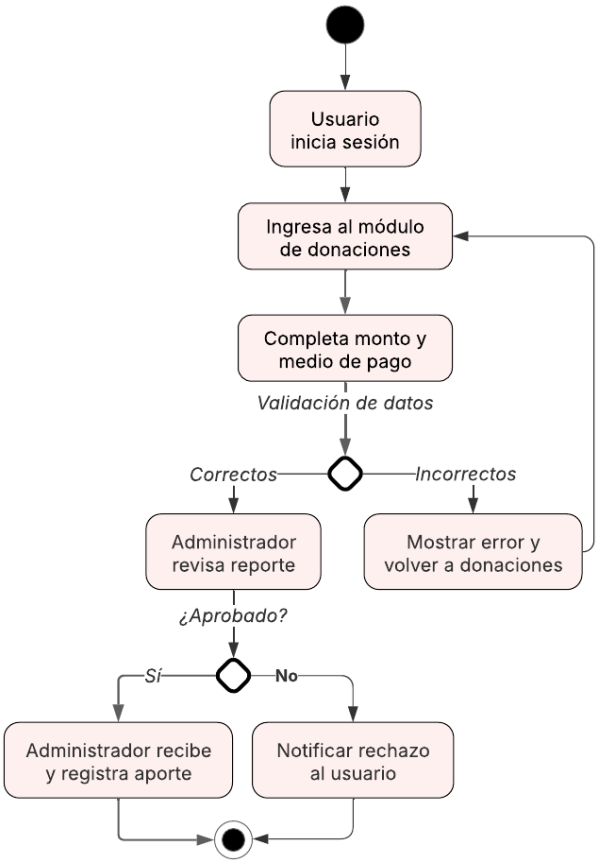
Proceso 1: Reportar mascota perdida

El proceso de reportar una mascota perdida comienza con la autenticación del usuario en el sistema. Una vez dentro, el usuario completa un formulario con la información básica de la mascota. El sistema valida los campos obligatorios y, posteriormente, un administrador revisa el reporte para garantizar la calidad de la información publicada. Si el reporte es aprobado, se publica en la plataforma y queda disponible para la comunidad; en caso contrario, se notifica al usuario. Este flujo asegura trazabilidad y organización en la gestión de mascotas perdidas.



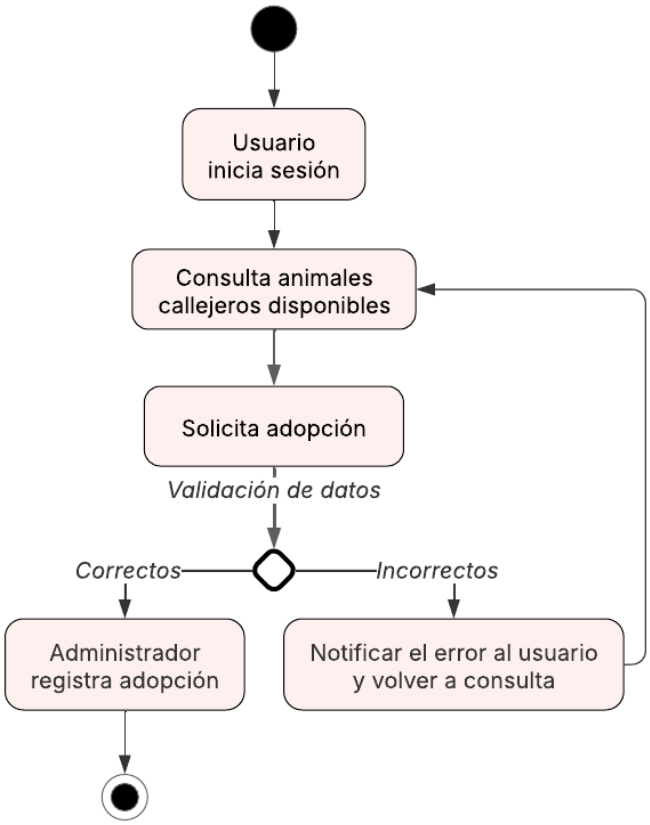
Proceso 2: Realizar una donación

El proceso de realizar una donación comienza cuando el usuario accede al módulo de donaciones dentro del sistema. Allí completa el monto que desea aportar y selecciona el medio de pago. Una vez confirmada la operación, el sistema genera automáticamente un comprobante de la transacción. Finalmente, el refugio recibe la información del aporte y lo registra en su base de datos, garantizando trazabilidad y control del ingreso recibido.



Proceso 3: Gestionar adopción

El proceso de gestión de adopción comienza cuando el usuario interesado consulta las mascotas disponibles en el sistema. Si desea una en particular, realiza la solicitud de adopción. El refugio se encarga de verificar la disponibilidad y las condiciones necesarias para concretar la entrega. Una vez aprobado, el administrador registra la adopción en el sistema, y la mascota cambia su estado a “adoptada”, asegurando un control formal y trazable del proceso.



Diagnóstico de procesos

Proceso 1: Reportar mascota perdida

- **Problema:** La difusión de mascotas perdidas se realiza principalmente en redes sociales de manera desorganizada, lo que provoca que los reportes se pierdan o no lleguen a las personas adecuadas.
- **Causas:** La falta de un registro centralizado y de un sistema que brinde trazabilidad de casos dificulta el seguimiento y la recuperación efectiva de las mascotas.

Proceso 2: Realizar una donación

- **Problema:** Las donaciones actualmente se gestionan de forma manual, lo que genera demoras y falta de transparencia en el control de aportes.
- **Causas:** No existe un sistema de registro automático ni un mecanismo confiable para la emisión de comprobantes, lo que limita la confianza y la trazabilidad de las transacciones.

Proceso 3: Gestionar adopción

- **Problema:** El registro de adopciones es informal y muchas veces se reduce a comunicaciones personales entre refugios y adoptantes, sin un seguimiento adecuado.
- **Causas:** La ausencia de una base de datos centralizada y la falta de un flujo estandarizado de gestión impiden garantizar un proceso formal, transparente y confiable para cada adopción.

Propuesta de solución

Propuesta funcional

El sistema informático para desarrollar tendrá como objetivo centralizar y optimizar la gestión de mascotas perdidas y animales callejeros en la ciudad de Córdoba.

De acuerdo con los alcances planteados, el sistema permitirá:

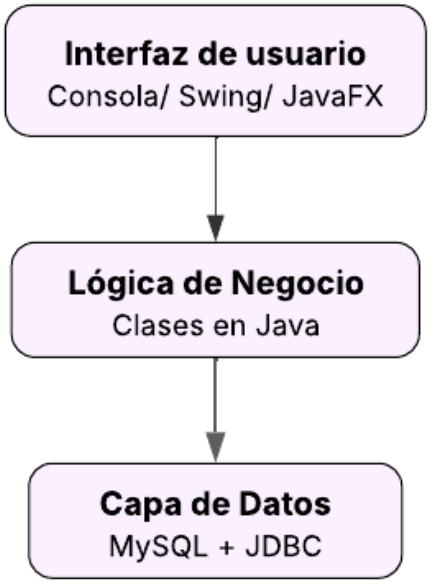
- Que los usuarios registren y consulten reportes de mascotas perdidas o encontradas.
- Que el refugio acceda a una gestión centralizada de los casos y de los aportes recibidos.
- Facilitar la difusión organizada de los casos dentro de la plataforma y, en una fase posterior, hacia redes sociales.
- Gestionar de forma transparente las donaciones comunitarias destinadas al refugio y a la atención de animales callejeros.
- Que los administradores supervisen, validen y mantengan actualizada la información registrada en el sistema.

Propuesta técnica

- **Lenguaje y entorno de desarrollo:** El sistema se implementará en Java, desarrollado en un entorno como Eclipse o IntelliJ IDEA.
- **Interfaz de usuario (prototipo):** Se utilizará una interfaz sencilla, que podrá ser implementada por consola, o con Swing / JavaFX, según la disponibilidad.
- **Base de datos:** Se utilizará MySQL para el almacenamiento persistente de datos. Acceso a datos mediante JDBC (Java Database Connectivity)**.**

Arquitectura del prototipo:

- **Capa de presentación:** interfaz por consola o GUI básica con formularios simples.
- **Capa de lógica de negocio:** clases en Java que gestionan la validación de datos y las reglas del sistema.
- **Capa de datos:** base MySQL con tablas para Usuarios, Mascotas, Reportes y Donaciones.



La arquitectura propuesta para el sistema se basa en un modelo en tres capas, lo que permite separar responsabilidades y garantizar una mejor organización del desarrollo. En la capa de presentación, los usuarios interactúan a través de una interfaz por consola o mediante Swing/JavaFX. La capa de lógica de negocio, desarrollada en Java, gestiona las reglas y validaciones del sistema. Finalmente, la capa de datos se implementa con MySQL, utilizando JDBC para el acceso y persistencia de la información.

Inicio del análisis

Fichas de Requerimientos Funcionales

ID: RF01 **Nombre:** Registro de usuarios **Tipo:** Funcional **Descripción:** Permitir que un usuario se registre con nombre, email y contraseña para poder acceder al sistema. **Entradas:** Nombre, email, contraseña. **Salidas:** Confirmación de registro exitoso y usuario creado en la base de datos. **Actores:** Usuario, Sistema. **Precondición:** Ninguna. **Postcondición:** Usuario registrado y habilitado para iniciar sesión. **Prioridad:** Alta **Dependencias:** Ninguna **Observaciones:** La contraseña debe cumplir políticas de seguridad básicas (mínimo 8 caracteres, combinación de letras y números).
ID: RF02 **Nombre:** Inicio y cierre de sesión **Tipo:** Funcional **Descripción:** Permitir a un usuario iniciar y cerrar sesión utilizando sus credenciales. **Entradas:** Email y contraseña. **Salidas:** Acceso permitido o denegado; sesión iniciada o cerrada. **Actores:** Usuario, Sistema. **Precondición:** Usuario registrado. **Postcondición:** Sesión del usuario iniciada o finalizada correctamente. **Prioridad:** Alta **Dependencias:** RF01 **Observaciones:** Se deben limitar intentos fallidos de acceso por seguridad.
ID: RF03 **Nombre:** Recuperación de contraseña **Tipo:** Funcional **Descripción:** Permitir que un usuario recupere su contraseña mediante un enlace o código enviado al email registrado. **Entradas:** Email del usuario registrado. **Salidas:** Enlace o código de recuperación enviado. **Actores:** Usuario, Sistema. **Precondición:** Usuario registrado. **Postcondición:** Usuario puede establecer una nueva contraseña. **Prioridad:** Media **Dependencias:** RF01 **Observaciones:** El enlace o código debe expirar después de un tiempo determinado.
ID: RF04 **Nombre:** Reportar mascota perdida **Tipo:** Funcional **Descripción:** Permitir que un usuario registre un reporte de mascota perdida incluyendo especie, descripción, fecha, ubicación y datos de contacto. **Entradas:** Especie, descripción, fecha, ubicación, datos de contacto, foto opcional. **Salidas:** Reporte registrado en la base de datos y notificación al administrador.

Actores: Usuario, Sistema, Administrador.
Precondición: Usuario registrado.
Postcondición: Reporte disponible para consulta y difusión.
Prioridad: Alta
Dependencias: RF01
Observaciones: La información puede ser compartida en la sección de reportes y redes sociales.

ID: RF05
Nombre: Reportar animal callejero/encontrado
Tipo: Funcional
Descripción: Permitir registrar animales encontrados o callejeros con los mismos campos que mascotas perdidas.
Entradas: Especie, descripción, fecha, ubicación, datos de contacto, foto opcional.
Salidas: Reporte registrado en la base de datos.
Actores: Usuario, Sistema, Administrador.
Precondición: Usuario registrado.
Postcondición: Reporte disponible para consulta y difusión.
Prioridad: Alta
Dependencias: RF01
Observaciones: Puede compartirse en tablero interno y redes sociales.

ID: RF06
Nombre: Adjuntar imagen a reporte
Tipo: Funcional
Descripción: Permitir que el usuario agregue una imagen opcional al reporte de mascota o animal callejero.
Entradas: Archivo de imagen (ruta o subida).
Salidas: Imagen asociada al reporte en la base de datos.
Actores: Usuario, Sistema.
Precondición: Usuario registrado y reporte creado.
Postcondición: Reporte actualizado con la imagen.
Prioridad: Media
Dependencias: RF04, RF05
Observaciones: Se validarán formatos y tamaño máximo del archivo.

ID: RF07
Nombre: Validación de datos obligatorios
Tipo: Funcional
Descripción: El sistema debe validar que los campos obligatorios estén completos antes de guardar un reporte.
Entradas: Datos del formulario del reporte.
Salidas: Confirmación de datos completos o mensaje de error.
Actores: Usuario, Sistema.
Precondición: Usuario registrado.
Postcondición: Reporte aceptado o rechazado hasta corregir datos faltantes.
Prioridad: Alta
Dependencias: RF04, RF05
Observaciones: Campos obligatorios: especie, fecha, zona.

ID: RF08
Nombre: Publicación de reportes en tablero
Tipo: Funcional

Descripción: Publicar reportes en un tablero interno, ordenado por fecha y con filtros.
Entradas: Reportes validados.
Salidas: Reportes visibles en tablero.
Actores: Sistema, Usuario, Administrador.
Precondición: Reporte validado.
Postcondición: Reporte accesible en el tablero para consulta.
Prioridad: Alta
Dependencias: RF04, RF05, RF07
Observaciones: Debe permitir filtros por especie, zona y estado.

ID: RF09
Nombre: Búsqueda y filtrado de reportes
Tipo: Funcional
Descripción: Permitir buscar y filtrar reportes por especie, estado, zona y fecha.
Entradas: Criterios de búsqueda.
Salidas: Lista de reportes que cumplen los criterios.
Actores: Usuario, Administrador.
Precondición: Reportes registrados.
Postcondición: Resultados filtrados mostrados correctamente.
Prioridad: Alta
Dependencias: RF08
Observaciones: Soporte para filtros combinados.

ID: RF10
Nombre: Ver detalle de reporte
Tipo: Funcional
Descripción: Permitir ver toda la información de un reporte, incluyendo datos de contacto del informante.
Entradas: Selección de reporte.
Salidas: Información completa del reporte.
Actores: Usuario, Administrador.
Precondición: Reporte registrado.
Postcondición: Información mostrada correctamente.
Prioridad: Alta
Dependencias: RF04, RF05
Observaciones: Solo administradores pueden ver datos sensibles completos.

ID: RF11
Nombre: Marcar reporte como resuelto
Tipo: Funcional
Descripción: Permitir marcar un reporte como resuelto y registrar la fecha correspondiente.
Entradas: Selección de reporte, tipo de resolución.
Salidas: Estado del reporte actualizado.
Actores: Usuario, Administrador.
Precondición: Reporte registrado.
Postcondición: Reporte con estado actualizado a resuelto.
Prioridad: Alta
Dependencias: RF04, RF05
Observaciones: Tipos de resolución: reencuentro, adopción, derivado al refugio.

ID: RF12
Nombre: Generar contenido de difusión para redes sociales

Tipo: Funcional
Descripción: Generar texto con los datos clave del reporte para compartir manualmente en redes sociales.
Entradas: Reporte validado.
Salidas: Contenido listo para publicación.
Actores: Administrador, Sistema.
Precondición: Reporte aprobado.
Postcondición: Contenido generado.
Prioridad: Media
Dependencias: RF04, RF05
Observaciones: Texto formateado con enlace/ID del caso.

ID: RF13
Nombre: Revisión y aprobación de reportes
Tipo: Funcional
Descripción: Permitir a administradores revisar, aprobar, editar o rechazar reportes antes de su publicación.
Entradas: Reporte pendiente.
Salidas: Reporte aprobado, editado o rechazado.
Actores: Administrador.
Precondición: Reporte registrado.
Postcondición: Reporte en estado final: aprobado, editado o rechazado.
Prioridad: Alta
Dependencias: RF04, RF05
Observaciones: Se notifica al usuario sobre la decisión.

ID: RF14
Nombre: Gestión de usuarios
Tipo: Funcional
Descripción: Permitir a administradores activar o bloquear cuentas de usuarios.
Entradas: Usuario seleccionado, acción a ejecutar.
Salidas: Estado de usuario actualizado.
Actores: Administrador.
Precondición: Usuario registrado.
Postcondición: Usuario activo o bloqueado según decisión.
Prioridad: Alta
Dependencias: RF01
Observaciones: Cambios notificados al usuario.

ID: RF15
Nombre: Registrar donación
Tipo: Funcional
Descripción: Permitir que usuarios registren una donación asociada al refugio.
Entradas: Usuario, monto, fecha, medio de pago.
Salidas: Donación registrada y confirmación enviada.
Actores: Usuario, Sistema.
Precondición: Usuario registrado.
Postcondición: Donación almacenada en base de datos.
Prioridad: Alta
Dependencias: RF01
Observaciones: Integración con API de pasarela de pago opcional.

ID: RF16
Nombre: Emitir comprobante de donación
Tipo: Funcional
Descripción: Permitir mostrar y guardar un comprobante simple de donación.
Entradas: Registro de donación.
Salidas: Comprobante en pantalla y opción de descarga.
Actores: Usuario, Sistema.
Precondición: Donación registrada.
Postcondición: Comprobante generado.
Prioridad: Media
Dependencias: RF15
Observaciones: Formato PDF opcional.

ID: RF17
Nombre: Consultar aportes recibidos
Tipo: Funcional
Descripción: Permitir listar y filtrar aportes por período.
Entradas: Filtro por fecha.
Salidas: Lista de aportes recibidos.
Actores: Administrador.
Precondición: Donaciones registradas.
Postcondición: Listado filtrado mostrado.
Prioridad: Media
Dependencias: RF15, RF16
Observaciones: Exportación opcional a Excel.

ID: RF18
Nombre: Trazabilidad de cambios de estado
Tipo: Funcional
Descripción: Mantener registro de quién y cuándo modifica el estado de un reporte.
Entradas: Modificación de reporte.
Salidas: Registro de auditoría en base de datos.
Actores: Usuario, Administrador, Sistema.
Precondición: Reporte registrado.
Postcondición: Registro de cambios actualizado.
Prioridad: Media
Dependencias: RF04, RF05, RF11
Observaciones: Información accesible solo a administradores.

Fichas de Requerimientos Candidatos (a futuro)

ID: RFC01
Nombre: Integración con APIs de redes sociales
Tipo: Candidato / Funcional
Descripción: Permitir la publicación automática de casos aprobados en redes sociales (Facebook e Instagram) para difundirlos de manera eficiente.
Entradas: Reporte aprobado.
Salidas: Publicación en redes sociales.
Actores: Administrador, Sistema
Precondición: Reporte aprobado; APIs activas y configuradas.
Postcondición: Reporte difundido automáticamente.

Prioridad: Media
Dependencias: RF12, RF13
Observaciones: Requiere conexión a internet y validación de credenciales de redes sociales.

ID: RFC02
Nombre: Integración con pasarela de pagos
Tipo: Candidato / Funcional
Descripción: Permitir procesar donaciones en línea de manera segura mediante la integración con una pasarela de pagos.
Entradas: Datos de donación, información de pago.
Salidas: Confirmación de donación procesada.
Actores: Usuario, Sistema
Precondición: Usuario registrado; pasarela de pagos operativa.
Postcondición: Donación registrada y confirmada.
Prioridad: Media
Dependencias: RF15, RF16
Observaciones: Requiere manejo seguro de datos financieros y cumplimiento de normativas de pago.

ID: RFC03
Nombre: Notificaciones automáticas
Tipo: Candidato / Funcional
Descripción: Enviar notificaciones a los usuarios (email/WhatsApp) cuando haya reportes cercanos o cambios de estado relevantes.
Entradas: Nuevo reporte, cambio de estado de reporte.
Salidas: Notificación enviada al usuario.
Actores: Usuario, Sistema
Precondición: Usuario registrado y con contacto válido.
Postcondición: Usuario informado oportunamente.
Prioridad: Media
Dependencias: RF04, RF05, RF11
Observaciones: Debe permitir suscripción/desuscripción de notificaciones.

ID: RFC04
Nombre: Geolocalización en mapa y radio de búsqueda
Tipo: Candidato / Funcional
Descripción: Permitir mostrar reportes en un mapa y buscar mascotas perdidas o animales callejeros dentro de un radio determinado.
Entradas: Ubicación del usuario, radio de búsqueda.
Salidas: Lista de reportes dentro del área definida.
Actores: Usuario, Sistema
Precondición: Usuario registrado; ubicación disponible.
Postcondición: Reportes filtrados y visualizados en mapa.
Prioridad: Media
Dependencias: RF04, RF05, RF09
Observaciones: Requiere conexión a internet y acceso a GPS o geolocalización.

ID: RFC05
Nombre: Dashboards con métricas
Tipo: Candidato / Funcional
Descripción: Generar dashboards con métricas sobre casos por zona/mes, montos donados y tiempos de resolución de reportes.

Entradas: Datos de reportes y donaciones.
Salidas: Dashboard interactivo con gráficos y estadísticas.
Actores: Administrador, Sistema
Precondición: Reportes y donaciones registrados.
Postcondición: Información visualizada y disponible para análisis.
Prioridad: Media
Dependencias: RF04, RF05, RF15, RF17
Observaciones: Permitir exportación de datos y filtros por fecha, zona o tipo de reporte.

Fichas de Requerimientos No Funcionales

ID: RNF01
Nombre: Usabilidad
Tipo: No funcional
Descripción: La interfaz debe ser simple, intuitiva y accesible para usuarios sin experiencia técnica.
Entradas: Interacción del usuario.
Salidas: Respuesta del sistema clara y entendible.
Actores: Usuario
Precondición: Usuario accede al sistema.
Postcondición: Operaciones ejecutadas correctamente.
Prioridad: Alta
Dependencias: Ninguna
Observaciones: Compatible con guías de accesibilidad.

ID: RNF02
Nombre: Rendimiento
Tipo: No funcional
Descripción: El sistema debe responder en pocos segundos a operaciones básicas como registrar y consultar reportes.
Entradas: Solicitud de operación.
Salidas: Resultado de operación.
Actores: Usuario, Sistema
Precondición: Sistema operativo.
Postcondición: Respuesta rápida a la operación solicitada.
Prioridad: Alta
Dependencias: Ninguna
Observaciones: Medir tiempos de respuesta durante pruebas.

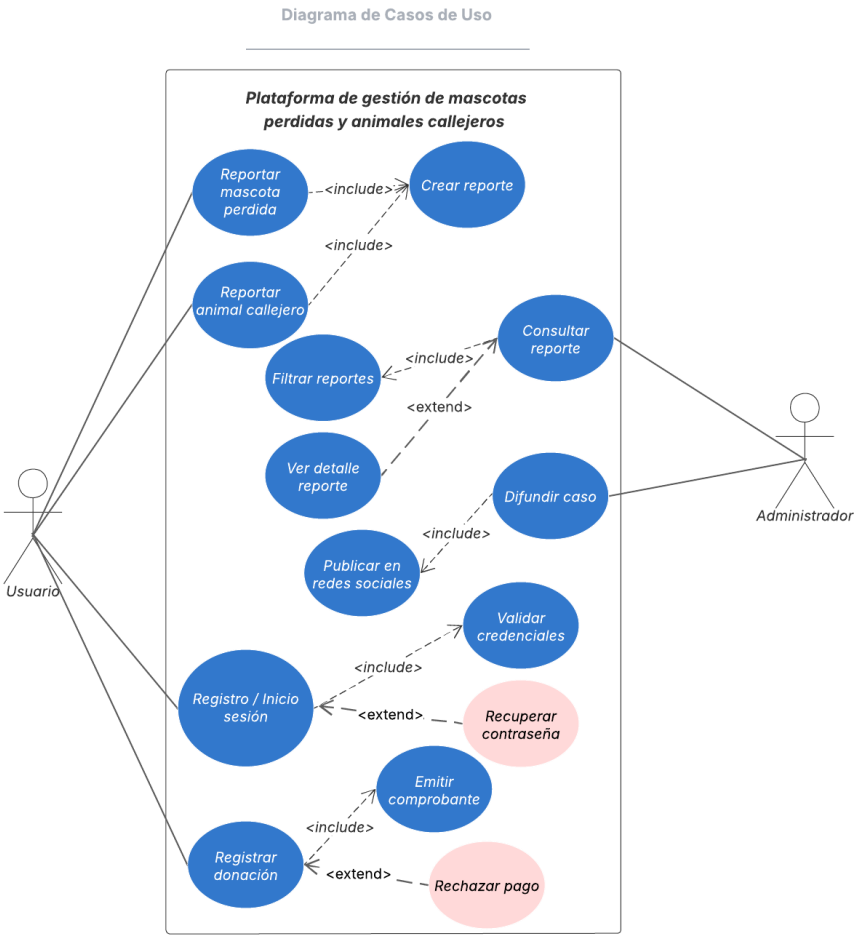
ID: RNF03
Nombre: Confiabilidad
Tipo: No funcional
Descripción: Asegurar la disponibilidad de la información mientras exista conexión a internet.
Entradas: Solicitudes de usuarios.
Salidas: Respuesta del sistema.
Actores: Usuario, Sistema
Precondición: Sistema en funcionamiento y conexión activa.
Postcondición: Información accesible sin errores.
Prioridad: Alta
Dependencias: Ninguna
Observaciones: Copias de seguridad periódicas.

ID: RNF04
Nombre: Seguridad

Tipo: No funcional
Descripción: Los datos de usuarios y donaciones deben estar protegidos mediante validación de accesos.
Entradas: Credenciales de acceso y acciones del usuario.
Salidas: Acceso permitido o denegado.
Actores: Usuario, Administrador, Sistema
Precondición: Usuario registrado.
Postcondición: Datos seguros y acceso controlado.
Prioridad: Alta
Dependencias: RF01, RF02
Observaciones: Implementar cifrado y políticas de contraseña segura.

ID: RNF05
Nombre: Portabilidad
Tipo: No funcional
Descripción: El sistema debe poder ejecutarse en cualquier PC que cuente con Java y MySQL instalados.
Entradas: Instrucción de ejecución.
Salidas: Sistema en funcionamiento.
Actores: Usuario, Sistema
Precondición: Entorno Java y MySQL disponibles.
Postcondición: Sistema operativo en cualquier equipo compatible.
Prioridad: Media
Dependencias: Ninguna
Observaciones: Probar en distintos sistemas operativos compatibles.

Diagrama de Casos de uso



El diagrama representa los casos de uso de una **plataforma de gestión de mascotas perdidas y animales callejeros**. Los actores principales son el **Usuario** y el **Administrador**.

El **Usuario** puede: registrar e iniciar sesión, reportar mascotas perdidas o animales callejeros, y registrar donaciones. Algunos casos de uso incluyen otros de manera obligatoria (<include>), como “Crear reporte” al reportar un animal o “Validar credenciales” al iniciar sesión.

El **Administrador** tiene la función de consultar reportes y difundir casos, interactuando con los mismos procesos que los usuarios para asegurar la correcta gestión de la información. Además, ciertos casos son opcionales (<extend>), como “Ver detalle reporte” al consultar un reporte.

Casos especiales, como **Recuperar contraseña** o **Rechazar pago**, representan situaciones excepcionales dentro del flujo principal de la plataforma.

Casos de uso seleccionados

- **CU001** Reportar mascota perdida
- **CU002** Difundir caso
- **CU003** Registrar donación

Tabla de trazabilidad

Requerimiento	**Caso de uso**	**Actor principal**	**Paquete del análisis**	**Comentario**
RF04. Reportar mascota perdida	**CU001**	Usuario	Reportes	Permite crear un reporte en la plataforma.
RF09. Consultar reportes de mascotas	**CU001**	Administrador	Reportes	Relacionado con la visualización de casos.
RF12. Difusión de casos en redes sociales	**CU002**	Administrador	Difusión	Publicación automática en Facebook e Instagram.
RF15. Gestionar donaciones comunitarias	**CU003**	Usuario	Donaciones	Pago electrónico integrado con API externa.
RF18. Supervisar y validar reportes	**CU001, CU002, CU003**	Administrador	Gestión	Revisión y autorización de la información cargada.

Especificación de casos de uso

Tabla CU001 – Reportar mascota perdida o encontrada

Elemento	Especificación
Actores	Usuario
Referencias	RF04, RF09, RF18
Descripción	Permite a un usuario registrar un reporte de mascota perdida o encontrada, con información de contacto, foto y ubicación.
Precondición	El usuario debe estar registrado o identificado.
Flujo principal	1. El usuario selecciona la opción "Reportar mascota". 2. El sistema despliega un formulario para cargar datos: tipo de reporte (perdida/encontrada), descripción, ubicación y foto. 3. El usuario completa los datos y confirma. 4. El sistema guarda la información en la base de datos. 5. El sistema notifica al administrador para su validación.

Tabla CU002 – Difundir caso en redes sociales

Elemento	Especificación
Actores	Administrador
Referencias	RF12, RF18
Descripción	Permite difundir de manera automática un reporte validado en las redes sociales vinculadas al sistema.
Precondición	El administrador validó el reporte. El sistema tiene conectores activos a las redes.
Flujo principal	1. El sistema identifica reportes aprobados. 2. El administrador selecciona los reportes a difundir. 3. El sistema genera automáticamente el contenido (texto + foto). 4. El sistema envía el reporte a Facebook e Instagram. 5. El sistema confirma la publicación exitosa.
Postcondición	El reporte queda publicado en las redes sociales vinculadas.
Flujo alternativo	• Si hay error en la conexión con la API, el sistema informa al administrador y reintenta luego.

Tabla CU003 – Realizar donación en línea

Elemento	Especificación
Actores	Usuario
Referencias	RF15, RF18
Descripción	Permite al usuario realizar un aporte monetario al refugio de forma electrónica.
Precondición	El usuario debe tener acceso a un medio de pago válido.
Flujo principal	1. El usuario selecciona "Donar". 2. El sistema muestra opciones de monto fijo y monto libre. 3. El usuario selecciona el monto y confirma. 4. El sistema redirige a la pasarela de pago (API). 5. La plataforma de pago procesa la operación y devuelve el resultado. 6. El sistema registra la donación en la base de datos. 7. El administrador recibe la confirmación de aporte. 8. El usuario recibe comprobante digital.
Postcondición	La donación queda registrada y acreditada.
Flujo alternativo	• Si el pago es rechazado, se informa al usuario y se cancela el proceso.

SEGUNDA ENTREGA

Objetivos

Esta actividad busca que seas capaz de plantear una solución problemática que pueda resolverse mediante la realización de un proyecto informático.

Para llevar adelante este desafío, vas a poder aplicar muchos de los conceptos más importantes abordados durante el desarrollo del módulo 2, que recupera tu trabajo en materias troncales de la carrera.

Durante el proceso, lograrás aplicar tus conocimientos para alcanzar los siguientes objetivos:

- Reconocer y aplicar el proceso unificado de desarrollo.
- Utilizar UML para representar y comunicar los aspectos del sistema.
- Diseñar un modelo de datos relacional y realiza un diagrama entidad-relación.
- Emplear consultas SQL para gestionar una base de datos.
- Plantear los requerimientos de comunicación del sistema.

En este segundo TP deberás retomar los requerimientos planteados para el sistema a desarrollar y avanzar con el proyecto.

Enunciado

Los sistemas informáticos desempeñan un papel muy importante para la optimización de procesos en diversas áreas de cualquier organización. La tecnología brinda la posibilidad de automatizar tareas, recopilar y analizar datos a gran escala, agilizar procesos y proporcionar soluciones a desafíos complejos.
Te propongo que retomes lo que definiste en la actividad anterior y realices una entrega de acuerdo con lo solicitado en la consigna. Recuerda que puedes definir cualquier organización real y explorar oportunidades en relación con su seguridad, logística, gestión de inventarios, optimización de procesos industriales, análisis de datos, cuidado de la salud, la educación, o cualquier otra área.
Para continuar con la realización de un proyecto de desarrollo informático, se requiere aplicar de forma completa los modelos de análisis, diseño, implementación y pruebas del PUD. Es clave elegir adecuadamente los artefactos a presentar y utilizar correctamente el lenguaje unificado de desarrollo (UML).
En cuanto a la persistencia de los datos, se requiere elaborar un modelo relacional que represente las entidades, sus atributos y las relaciones específicas propias al sistema. Se deben aplicar los principios de normalización con el fin de garantizar que las tablas en la base de datos estén organizadas de manera eficaz y presentar un diagrama de entidad relación.
Paralelamente, se evalúan los requisitos de comunicación del sistema y se determina cómo se llevan a cabo las interacciones entre los diversos componentes del mismo.

Consignas

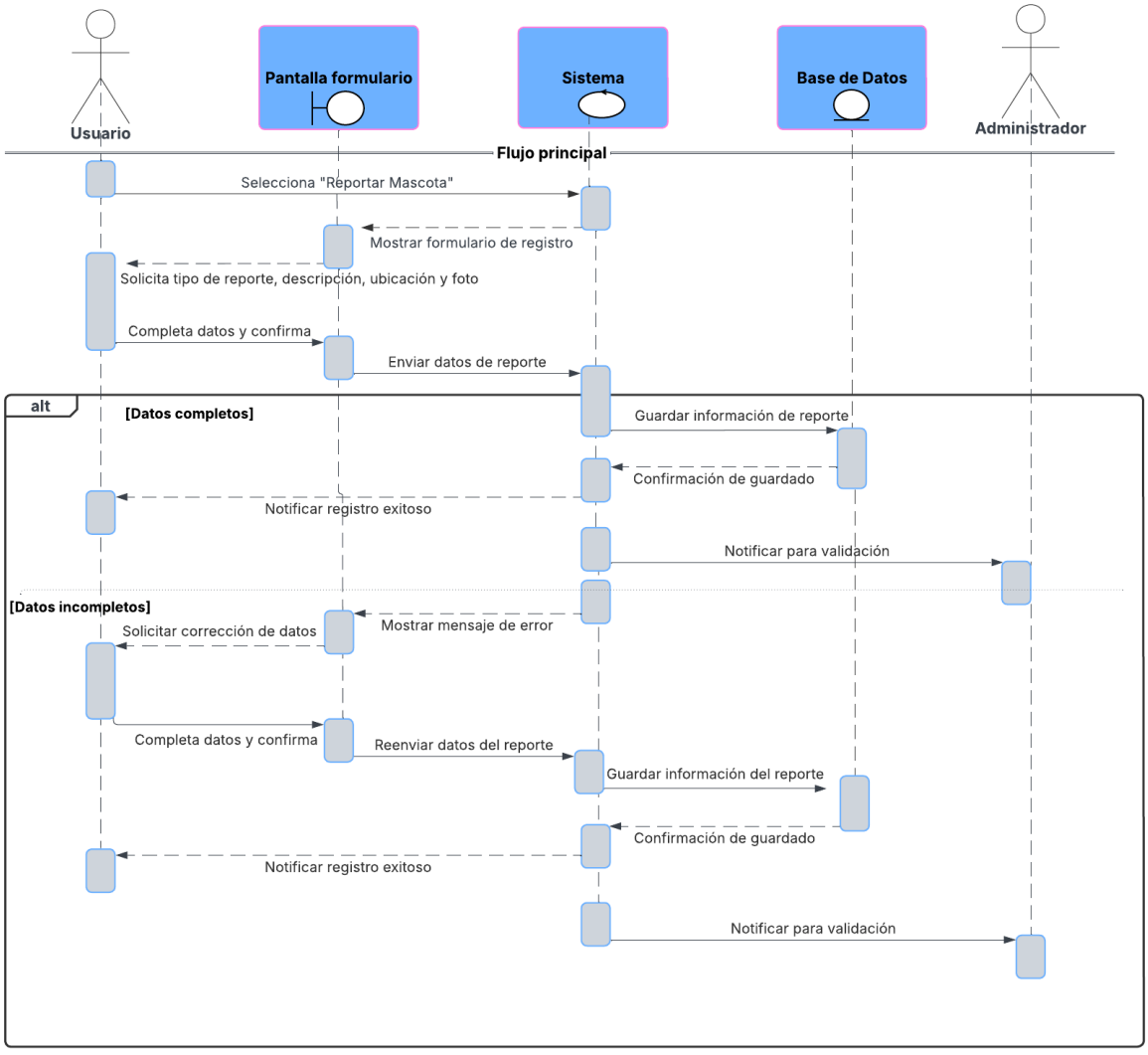
- Etapa de análisis.
- Etapa de diseño.
- Etapa de implementación.
- Etapa de pruebas.
- Definición de base de datos para el sistema.
- Diagrama entidad-relación de la base de datos.
- Creación de las tablas MySQL.
- Inserción, consulta y borrado de registros.
- Presentación de las consultas SQL.
- Definiciones de comunicación.

Etapa de Análisis

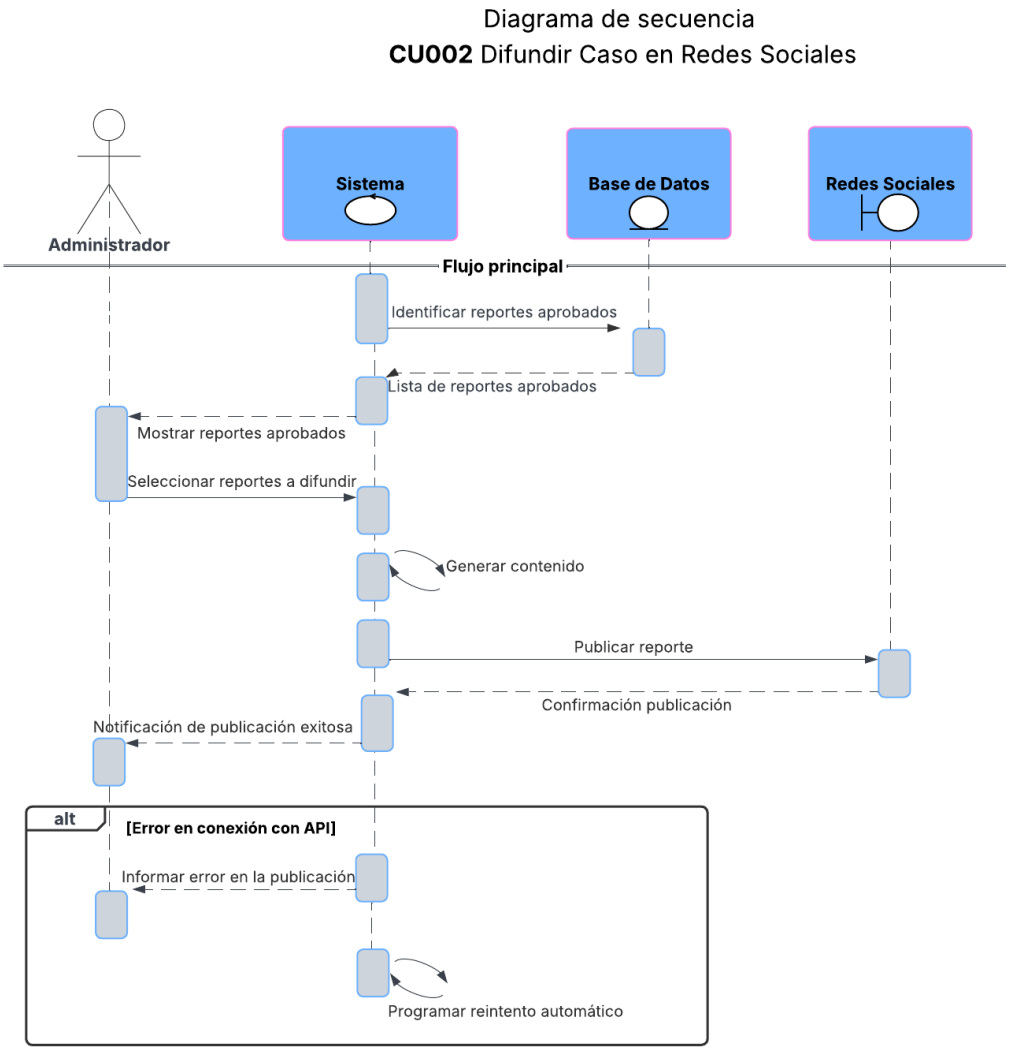
A continuación, se presentan los diagramas de secuencia correspondientes a las especificaciones de los tres casos de uso del sistema, detallados anteriormente.

- **CU001 – Registrar reporte de mascota**: muestra la interacción entre el usuario, la interfaz de formulario, el sistema, la base de dato y el administrador para registrar un reporte de mascota perdida o encontrada. Incluye además un flujo alternativo(alt) en caso de que los datos ingresados sean incompletos.

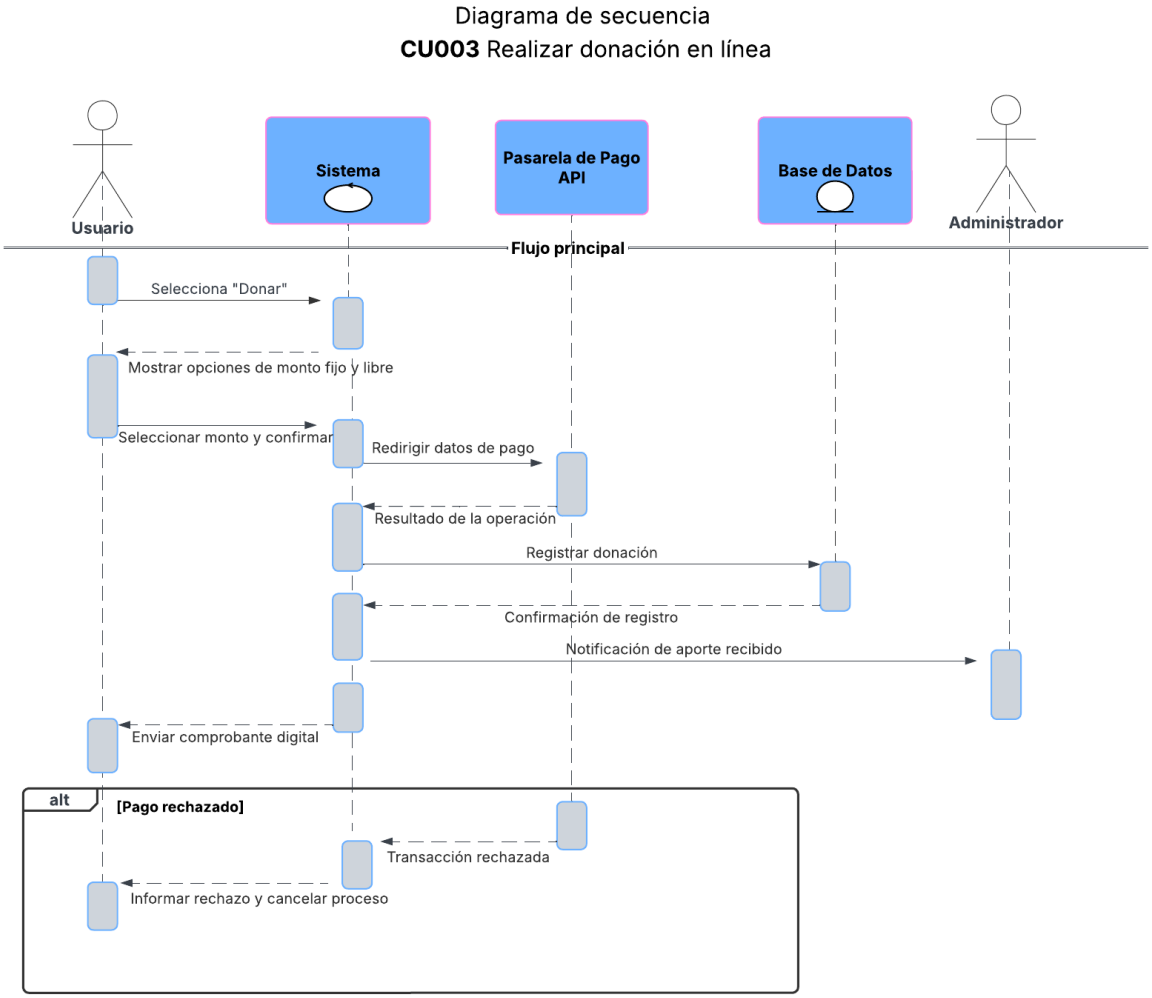
Diagrama de secuencia
CU001 Reportar mascota perdida



- **CU002 – Difundir caso en redes sociales**: refleja cómo el sistema y el administrador interactúan para publicar reportes previamente validados en las redes sociales vinculadas. Se contempla también un flujo alternativo en caso de error en la conexión con la API.

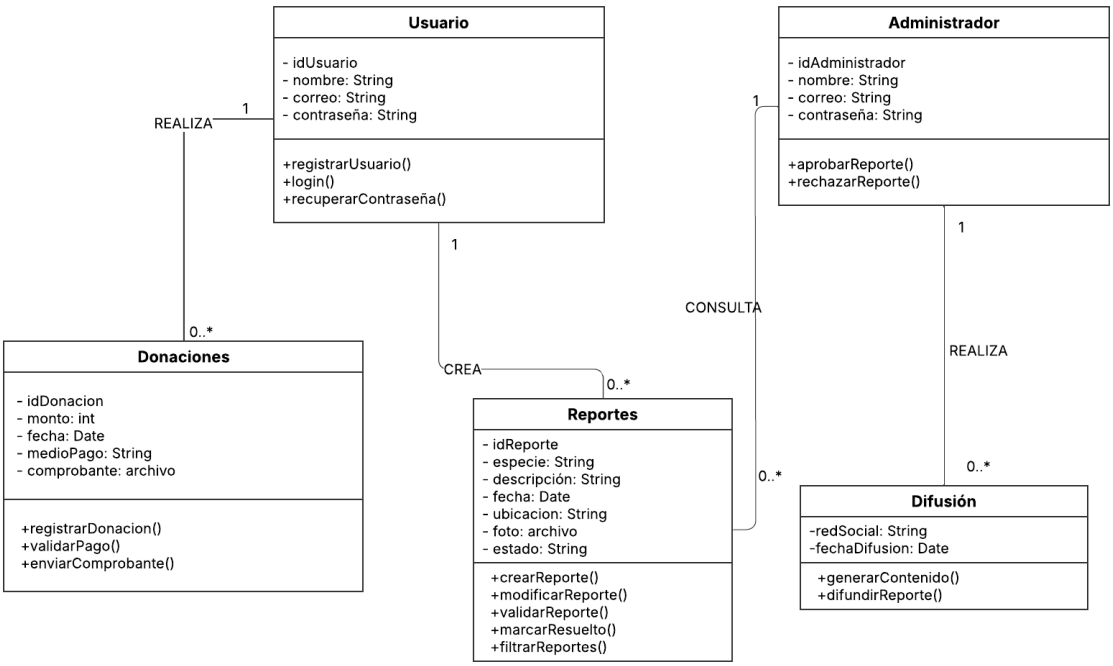


- **CU003 – Realizar donación en línea**: describe el proceso mediante el cual un usuario realiza una donación a través de una pasarela de pago externa, incluyendo el registro en la base de datos y la notificación al administrador. El flujo alternativo contempla la situación de un pago rechazado.



Estos diagramas permiten visualizar de manera clara las interacciones entre los actores, las entidades y los sistemas externos involucrados en cada caso de uso, destacando tanto el flujo principal como las excepciones que pueden ocurrir.

Se adjunta el correspondiente diagrama de clases



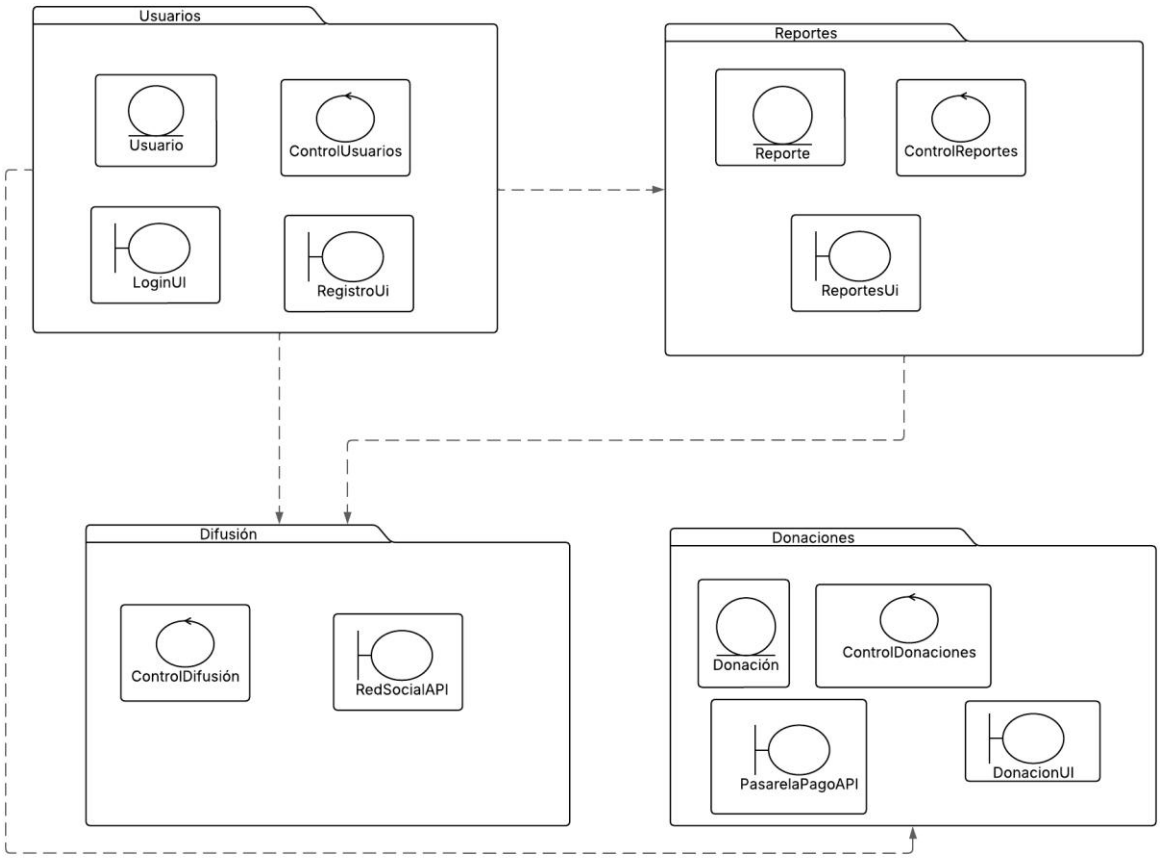
El diagrama representa la estructura del sistema, mostrando las clases principales, sus atributos, métodos y relaciones.

- **Usuario**: puede registrarse, iniciar sesión y recuperar su contraseña. Se relaciona con las donaciones que realiza y con los reportes que crea.
- **Donaciones**: registran la información de cada aporte (monto, fecha, medio de pago, comprobante), incluyendo funciones para validar y enviar comprobantes.
- **Reportes**: contienen información sobre mascotas perdidas o encontradas (especie, descripción, ubicación, foto, estado). Pueden ser creados y gestionados por los usuarios, y son consultados por los administradores.
- **Administrador**: gestiona los reportes creados, pudiendo aprobarlos o rechazarlos.
- **Difusión**: permite generar y publicar contenido en redes sociales, difundiendo los reportes aprobados.

En cuanto a las **relaciones**:

- Un usuario puede realizar múltiples donaciones y generar varios reportes.
- Los administradores consultan los reportes creados por usuarios y deciden su aprobación.
- Una vez aprobado, el reporte puede difundirse en distintas redes sociales.

Luego, el correspondiente paquete de análisis



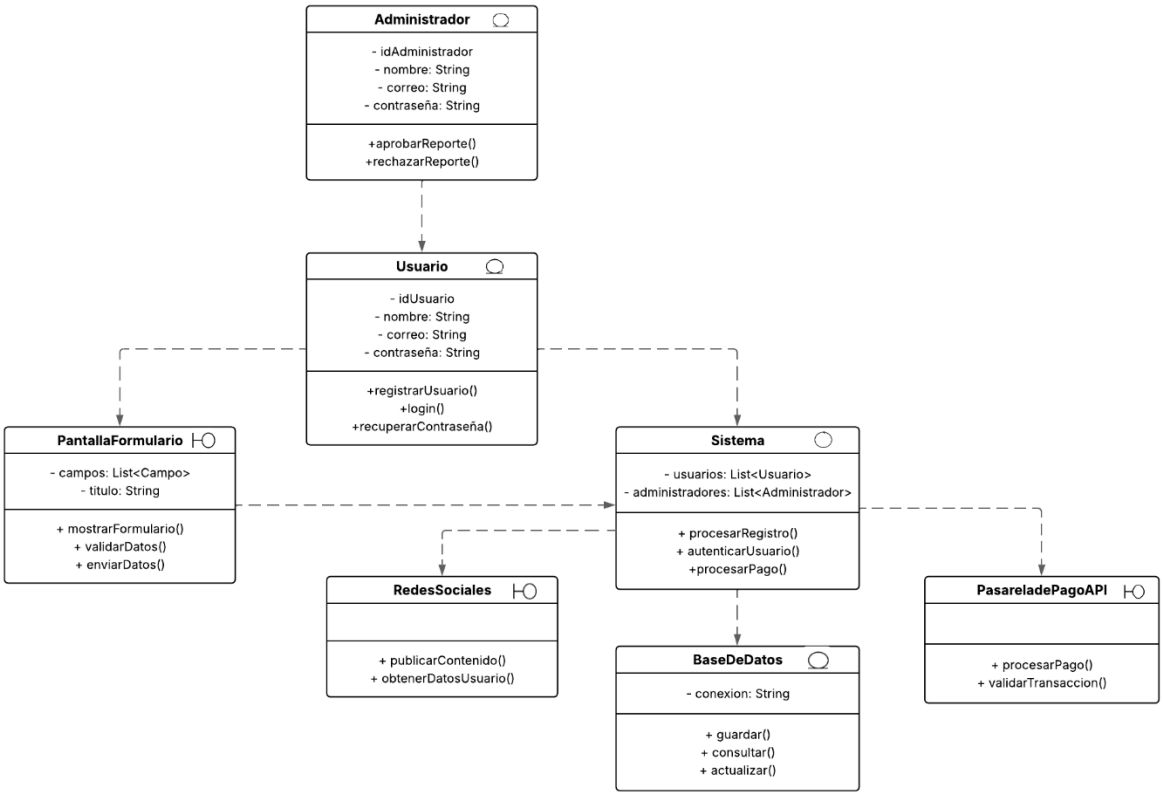
El diagrama presenta la organización del sistema en distintos paquetes que agrupan clases y componentes relacionados por funcionalidad.

- **Usuarios:** Contiene las clases e interfaces relacionadas con la gestión de usuarios, incluyendo el registro, el login y el control general de usuarios.
- **Reportes:** Agrupa los elementos para la generación y visualización de reportes, así como el control de los mismos.
- **Difusión:** Incluye componentes para la difusión de información a través de redes sociales, manejados por el controlador de difusión y la API de la red social.
- **Donaciones:** Contiene las clases vinculadas al manejo de donaciones, su control, la interfaz de usuario y la integración con la pasarela de pagos.

Las flechas indican dependencias entre paquetes, mostrando cómo cada módulo puede interactuar con otros para cumplir con los flujos de información del sistema. Esto permite una visión estructurada y modular del sistema, facilitando su análisis y posterior implementación.

Etapa de diseño

A continuación, se presenta el diagrama de clases de diseño, este se construyó a partir del análisis detallado de los diagramas de secuencia correspondientes a los casos de uso.



- **CU001 – Registrar reporte de mascota perdida o encontrada**,
- **CU002 – Difundir caso en redes sociales**, y
- **CU003 – Realizar donación en línea**.

Estos casos de uso describen las interacciones entre los usuarios y el sistema para registrar reportes de mascotas, compartir información en redes sociales y procesar donaciones en línea.

A partir de esta especificación, se identificaron las clases necesarias para cumplir con los requerimientos funcionales, clasificándolas en:

Entidades, que representan objetos del dominio con datos persistentes:

- **Usuario**
- **Administrador**
- **BaseDeDatos**

Controles, que encapsulan la lógica de negocio:

- **Sistema**

Interfaces, que manejan la interacción con usuarios o servicios externos:

- **FormularioPantalla**
- **RedesSociales**
- **PasarelaPagoAPI**

Esta organización asegura una **separación clara de responsabilidades**, facilitando la comprensión y mantenimiento del sistema. Cada clase cumple un rol específico dentro de los casos de uso, reflejando fielmente las necesidades definidas en los diagramas de secuencia y garantizando la correcta interacción entre los usuarios, el sistema y los servicios externos.

Etapa de implementación

La etapa de implementación busca definir cómo se desplegará el sistema en una infraestructura física y lógica. Para ello se utiliza un **diagrama de despliegue UML**, el cual muestra la distribución de los componentes de software en los nodos físicos (cliente, servidor de aplicaciones, servidor de base de datos) y sus interacciones.

Requerimientos considerados

Requerimientos Funcionales principales que influyen en la implementación:

- RF01: Registro de usuarios
- RF02: Inicio y cierre de sesión
- RF04: Reportar mascota perdida
- RF05: Reportar animal callejero/encontrado
- RF08: Publicación de reportes en tablero
- RF09: Búsqueda y filtrado de reportes
- RF15: Registrar donación
- RF16: Emitir comprobante de donación

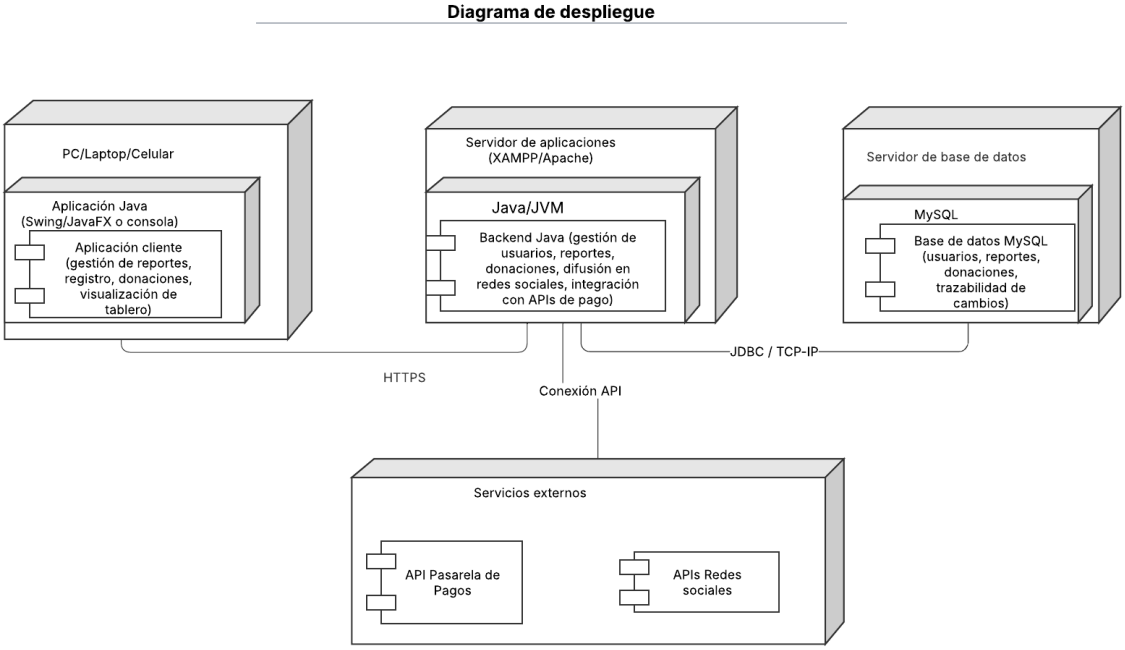
Requerimientos No Funcionales aplicables:

- RNF01: Usabilidad (interfaz simple con Swing/JavaFX)
- RNF02: Rendimiento (respuestas rápidas a operaciones básicas)
- RNF03: Confiabilidad (acceso estable a información con conexión a internet)
- RNF04: Seguridad (validación de accesos, políticas de contraseñas, cifrado)
- RNF05: Portabilidad (ejecución en PCs con Java y MySQL)

Tecnologías utilizadas

- **Java** para la lógica del sistema.
- **Swing/JavaFX** o consola para la interfaz del prototipo.
- **MySQL** como base de datos relacional.
- **JDBC** para la conexión aplicación–base de datos.
- **XAMPP** como entorno de desarrollo (Apache + MySQL + PHPMyAdmin).
- **GitHub** como repositorio para control de versiones.
- **APIs de pago electrónico** para gestión de donaciones (futuro: integración directa).
- **Conectores para redes sociales** (difusión manual, y futura integración automática).

Diagrama de Despliegue



Nodos físicos y componentes:

Cliente (PC/Laptop/Celular del usuario)

Navegador web o aplicación Java con Swing/JavaFX.

Servidor de Aplicaciones (XAMPP/Apache)

- Backend en Java (gestión de usuarios, reportes, donaciones).
- Módulo de integración con APIs de pago.
- Módulo de generación de contenido para difusión en redes sociales.

Servidor de Base de Datos (MySQL)

Almacenamiento de usuarios, reportes, donaciones, trazabilidad de cambios.

Conexiones:

- Cliente ↔ Servidor de Aplicaciones (HTTP/HTTPS).
- Servidor de Aplicaciones ↔ Servidor de Base de Datos (conexión JDBC).
- Servidor de Aplicaciones ↔ APIs externas (pasarela de pagos, redes sociales).

El sistema sigue una **arquitectura cliente-servidor**. El cliente accede a través de una aplicación Java, que envía solicitudes al servidor de aplicaciones desplegado en XAMPP. Este servidor gestiona la lógica de negocio en Java y se comunica con la base de datos MySQL mediante JDBC. Además, el servidor puede conectarse a APIs externas para el registro de donaciones y difusión en redes sociales.

La arquitectura planteada permite:

- **Seguridad y confiabilidad** mediante validaciones y cifrado de datos.
- **Escalabilidad** al separar la lógica de negocio de la base de datos y las integraciones.

- **Portabilidad** gracias al uso de Java y MySQL en cualquier PC con entorno compatible.

Etapa de pruebas

Durante este flujo de trabajo, se planifican y ejecutan las pruebas del sistema, tanto en versiones intermedias como en la final. Se consideran los siguientes artefactos: **plan de prueba, modelo de pruebas (casos, procedimientos y componentes), tratamiento de defectos y evaluación de pruebas**.

1. Plan de Prueba

- **Objetivo**: Verificar que el sistema cumpla los requerimientos funcionales y no funcionales definidos.
- **Alcance**: Se probarán los casos de uso principales:
    - CU001: Reportar mascota perdida.
    - CU002: Difundir caso validado en redes sociales.
    - CU003: Registrar donación.
- Criterios de aceptación:
    - Todos los casos de uso deben ejecutarse sin errores críticos.
    - El 95% de los casos de prueba deben aprobarse.
- Recursos:
    - Cliente (PC/Celular con navegador).
    - Servidor de aplicaciones (Java + Apache).
    - Servidor de base de datos (MySQL).
- **Responsables**: Equipo de QA y desarrolladores.

2. Modelo de pruebas

Casos de prueba

ID	Caso de uso	Caso de prueba	Entrada	Resultado esperado
CP001	CU001	Reportar mascota perdida con datos completos	Datos correctos + foto	Reporte almacenado, confirmación y notificación a admin
CP002	CU001	Reportar mascota sin ubicación	Falta campo ubicación	Error y bloqueo de guardado
CP003	CU002	Difundir caso validado en redes sociales	Caso validado y aprobado	Publicación exitosa en Facebook/Twitter
CP004	CU002	Difundir caso con conexión caída	Conexión simulada caída	Error + reintento posterior

ID	Caso de uso	Caso de prueba	Entrada	Resultado esperado
CP005	CU003	Registrar donación válida	Datos de pago correctos	Donación registrada y comprobante generado
CP006	CU003	Registrar donación con pago rechazado	Datos de tarjeta inválidos	Rechazo informado, no se registra la donación

3. Procedimientos de prueba

CU001 – Reportar mascota perdida

1. Ingresar al sistema como usuario registrado.
2. Seleccionar opción “Reportar mascota”.
3. Completar formulario con todos los datos.
4. Confirmar registro.
5. Verificar que:
    - Se guardan datos en la base.
    - Aparece mensaje de confirmación.
    - Admin recibe notificación.

CU002 – Difundir caso validado

1. Ingresar al sistema como administrador.
2. Aprobar un reporte válido.
3. Seleccionar opción “Difundir en redes sociales”.
4. Confirmar difusión.
5. Verificar que:
    - El caso se publica correctamente.
    - Ante falla de conexión, se muestre error y se reprograme difusión.

CU003 – Registrar donación

1. Ingresar al sistema como usuario.
2. Seleccionar opción “Donar”.
3. Ingresar datos de pago válidos.
4. Confirmar operación.
5. Verificar que:

- La transacción se registre en la base.
- Se genere comprobante en PDF.
- Ante error de pago, se informe y no quede registro.

4. Evaluación de pruebas

ID Caso	Descripción	Resultado esperado	Resultado obtenido	Estado	Observaciones
CP001	Reportar mascota con datos completos	Registro correcto y notificación	OK	Aprobado	
CP002	Reportar mascota sin ubicación	Error bloqueante	OK	Aprobado	
CP003	Difundir caso validado	Publicación exitosa	Error (API caída)	Rechazado	Reintentar luego
CP004	Difundir caso con conexión caída	Error + reintento	OK	Aprobado	
CP005	Registrar donación válida	Donación registrada + comprobante	OK	Aprobado	
CP006	Registrar donación con pago rechazado	Rechazo informado	OK	Aprobado	

5. Tratamiento de defectos

ID Defecto	Caso asociado	Descripción	Severidad	Responsable	Estado
D001	CP003	Fallo en publicación automática en redes sociales (API caída)	Alta	Backend	Abierto
D002	CP001	Imagen no se guarda si excede 5MB	Media	Frontend	En análisis
D003	CP005	El comprobante no descarga en PDF	Baja	Backend	Corregido

Prototipos de interfaz gráfica

A continuación, se presentan los prototipos de interfaz correspondientes a los tres casos de uso principales definidos en el sistema. Estas representaciones permiten visualizar de manera preliminar cómo interactuarán los usuarios y administradores con la aplicación, manteniendo la trazabilidad con los requerimientos funcionales y los casos de uso definidos:

- **CU001 – Reportar mascota perdida:** formulario para que el usuario cargue los datos de la mascota y genere un reporte.
- **CU002 – Difundir caso en redes sociales:** interfaz que permite al administrador seleccionar un reporte validado y difundirlo automáticamente en las redes vinculadas.
- **CU003 – Realizar donación en línea:** pantalla destinada a que el usuario seleccione un monto y realice el pago electrónico, con confirmación y generación de comprobante.

Estos prototipos tienen como objetivo anticipar la experiencia de usuario (UX) y servir de guía para la etapa de diseño e implementación.

Definición de base de datos para el sistema.

Para el prototipo del sistema de gestión de reportes de mascotas perdidas y animales callejeros, se diseñó una base de datos relacional en **MySQL**.

La base de datos garantiza la persistencia de la información clave, incluyendo **usuarios, reportes, donaciones, difusiones y auditorías de cambios**, lo que permite mantener un historial detallado de las actividades realizadas en la plataforma.

Las principales entidades identificadas son:

- **Usuario**, que representa a las personas registradas en el sistema, incluyendo administradores.
- **Reporte**, que almacena la información sobre mascotas perdidas, encontradas o animales callejeros.
- **Donación**, que registra los aportes monetarios de los usuarios al refugio.
- **Difusión**, que permite asociar reportes aprobados a publicaciones en redes sociales.
- **Auditoría**, que mantiene la trazabilidad de los cambios de estado en los reportes.

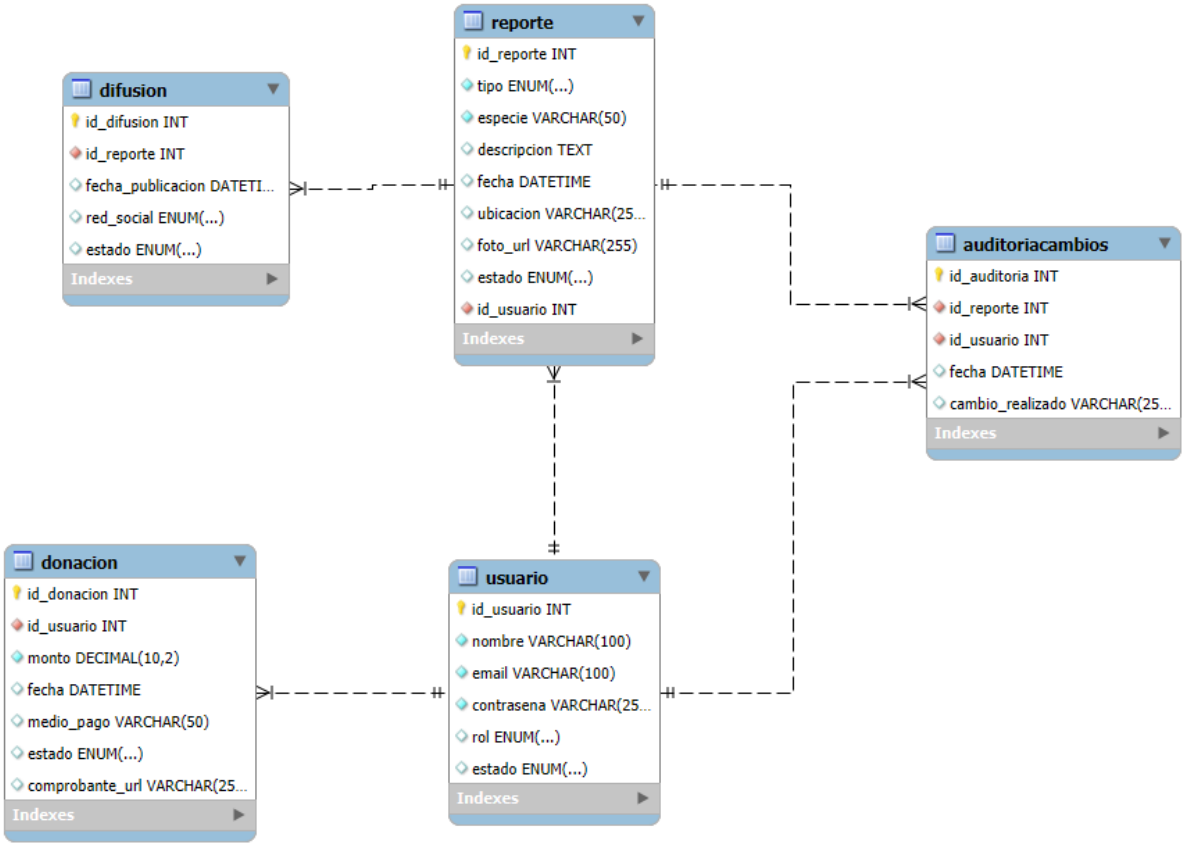
Estas entidades están vinculadas entre sí mediante relaciones de uno a muchos, lo que permite garantizar la integridad de los datos. Por ejemplo, un **usuario** puede generar múltiples reportes y donaciones, mientras que un **reporte** puede ser difundido en varias redes sociales.

Este esquema proporciona las bases necesarias para el prototipo, facilitando consultas, validaciones y el posterior escalamiento del sistema hacia funcionalidades más complejas, como integración con APIs de redes sociales o pasarelas de pago.

Diagrama entidad-relación de la base de datos

El siguiente Diagrama Entidad-Relación (DER) representa la estructura de la base de datos diseñada para la gestión y difusión de mascotas perdidas y animales callejeros (perros y gatos) en la ciudad de Córdoba Capital.

Se identifican las entidades principales: usuario, reporte, donación, difusión y auditoría de cambios, así como sus atributos clave y las relaciones entre ellas. Este modelo permite registrar la información de los usuarios, los reportes de mascotas perdidas o encontradas, las donaciones realizadas, la difusión en redes sociales y el seguimiento de cambios realizados en los reportes, garantizando la integridad y trazabilidad de los datos del sistema.



En cuanto a la **persistencia de los datos**, se utilizó una base de datos **MySQL**, diseñada a partir del diagrama entidad-relación desarrollado previamente. Las tablas fueron creadas aplicando los principios de normalización e integridad referencial, garantizando la coherencia y eficiencia del almacenamiento de la información.

Dentro del repositorio del proyecto se incluyen los archivos **.sql** con la definición completa de la base de datos, que contiene las sentencias **CREATE TABLE**, **INSERT**, **SELECT** y **DELETE** utilizadas para la gestión de los registros.

Este informe PDF, junto con el archivo .sql y los recursos asociados al prototipo, se encuentran disponibles en el siguiente repositorio de GitHub:

https://github.com/Florencia-otero1/Seminario-Siglo.git

Creación de las tablas MySQL

```sql
-- Tabla de usuarios
CREATE TABLE Usuario (
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    contrasena VARCHAR(255) NOT NULL,
    rol ENUM('usuario', 'administrador', 'voluntario') DEFAULT 'usuario',
    estado ENUM('activo', 'bloqueado') DEFAULT 'activo'
);

-- Tabla de reportes de mascotas
CREATE TABLE Reporte (
    id_reporte INT AUTO_INCREMENT PRIMARY KEY,
    tipo ENUM('perdida', 'encontrada', 'callejero') NOT NULL,
    especie VARCHAR(50) NOT NULL,
    descripcion TEXT,
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
    ubicacion VARCHAR(255),
    foto_url VARCHAR(255),
    estado ENUM('pendiente', 'validado', 'difundido', 'resuelto') DEFAULT 'pendiente',
    id_usuario INT NOT NULL,
    FOREIGN KEY (id_usuario)
        REFERENCES Usuario (id_usuario)
);
```

```sql
    -- Tabla de difusiones en redes sociales
CREATE TABLE Difusion (
    id_difusion INT AUTO_INCREMENT PRIMARY KEY,
    id_reporte INT NOT NULL,
    fecha_publicacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    red_social ENUM('Facebook', 'Instagram'),
    estado ENUM('exitoso', 'error') DEFAULT 'exitoso',
    FOREIGN KEY (id_reporte)
        REFERENCES Reporte (id_reporte)
);

    -- Tabla de donaciones
CREATE TABLE Donacion (
    id_donacion INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT NOT NULL,
    monto DECIMAL(10 , 2 ) NOT NULL,
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
    medio_pago VARCHAR(50),
    estado ENUM('aprobada', 'rechazada', 'pendiente') DEFAULT 'pendiente',
    comprobante_url VARCHAR(255),
    FOREIGN KEY (id_usuario)
        REFERENCES Usuario (id_usuario)
);

-- Tabla de auditoría de cambios
CREATE TABLE AuditoriaCambios (
    id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
    id_reporte INT NOT NULL,
    id_usuario INT NOT NULL,
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
    cambio_realizado VARCHAR(255),
    FOREIGN KEY (id_reporte)
        REFERENCES Reporte (id_reporte),
    FOREIGN KEY (id_usuario)
        REFERENCES Usuario (id_usuario)
);
```

Inserción, consulta y borrado de registros

Inserción de datos

```sql
-- Tabla Usuario
INSERT INTO Usuario (nombre, email, contrasena, rol, estado) VALUES
('Ana Pérez', 'ana.perez@gmail.com', 'contra1', 'usuario', 'activo'),
('Juan López', 'juan.lopez@gmail.com', 'contra2', 'administrador', 'activo'),
('María Gómez', 'maria.gomez@gmail.com', 'contra3', 'voluntario', 'bloqueado');

-- Tabla Reporte
INSERT INTO Reporte (tipo, especie, descripcion, fecha, ubicacion, foto_url, estado, id_usuario) VALUES
('perdida', 'perro', 'Perro marrón con collar rojo, perdido en plaza central.', '2025-09-15 14:30:00', 'Plaza Central, Ciudad', 'https://example.com/foto1.jpg', 'pendiente', 1),
('encontrada', 'gato', 'Gato blanco encontrado en la calle San Martín.', '2025-09-20 10:15:00', 'Calle San Martín 123', 'https://example.com/foto2.jpg', 'validado', 2),
('callejero', 'perro', 'Cachorro abandonado cerca de la terminal de ómnibus.', '2025-09-25 18:45:00', 'Terminal de ómnibus, Ciudad', 'https://example.com/foto3.jpg', 'difundido', 3);

-- Tabla Difusion
INSERT INTO Difusion (id_reporte, fecha_publicacion, red_social, estado) VALUES
(1, '2025-09-16 09:00:00', 'Facebook', 'exitoso'),
(2, '2025-09-21 15:30:00', 'Instagram', 'exitoso'),
(3, '2025-09-26 12:45:00', 'Facebook', 'error');

-- Tabla Donacion
INSERT INTO Donacion (id_usuario, monto, fecha, medio_pago, estado, comprobante_url) VALUES
(1, 1500.00, '2025-09-17 11:20:00', 'Tarjeta de crédito', 'aprobada', 'https://example.com/comprobante1.pdf'),
(2, 500.50, '2025-09-22 16:10:00', 'MercadoPago', 'pendiente', 'https://example.com/comprobante2.pdf'),
(3, 250.00, '2025-09-27 09:45:00', 'Transferencia bancaria', 'rechazada', 'https://example.com/comprobante3.pdf');

-- Tabla AuditoriaCambios
INSERT INTO AuditoriaCambios (id_reporte, id_usuario, fecha, cambio_realizado) VALUES
(1, 2, '2025-09-16 10:00:00', 'Actualizó la descripción del reporte.'),
(2, 1, '2025-09-21 16:00:00', 'Cambió el estado a validado.'),
(3, 3, '2025-09-26 13:00:00', 'Adjuntó foto del animal.');
```

Consultas

```sql
-- Ver todos los registros de auditoría
SELECT * FROM AuditoriaCambios;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

id_auditoria	id_reporte	id_usuario	fecha	cambio_realizado
4	1	2	2025-09-16 10:00:00	Actualizó la descripción del reporte.
5	2	1	2025-09-21 16:00:00	Cambió el estado a validado.
6	3	3	2025-09-26 13:00:00	Adjuntó foto del animal.
NULL	NULL	NULL	NULL	NULL

```sql
-- Ver todas las donaciones
SELECT * FROM Donacion;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_donacion	id_usuario	monto	fecha	medio_pago	estado	comprobante_url
7	1	1500.00	2025-09-17 11:20:00	Tarjeta de crédito	aprobada	https://example.com/comprobante1.pdf
8	2	500.50	2025-09-22 16:10:00	MercadoPago	pendiente	https://example.com/comprobante2.pdf
9	3	250.00	2025-09-27 09:45:00	Transferencia bancaria	rechazada	https://example.com/comprobante3.pdf
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```sql
-- Ver todas las difusiones en redes sociales
SELECT * FROM Difusion;
```

Result Grid | Filter Rows: | Edit: | Exp

id_difusion	id_reporte	fecha_publicacion	red_social	estado
1	1	2025-09-16 09:00:00	Facebook	exitoso
2	2	2025-09-21 15:30:00	Instagram	exitoso
3	3	2025-09-26 12:45:00	Facebook	error
NULL	NULL	NULL	NULL	NULL

```
101     -- Ver todos los reportes de mascotas
102 •   SELECT * FROM Reporte;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_reporte	tipo	especie	descripcion	fecha	ubicacion	foto_url	estado	id_usuario
1	perdida	perro	Perro marrón con collar rojo, perdido en plaza c...	2025-09-15 14:30:00	Plaza Central, Ciudad	https://example.com/foto1.jpg	pendiente	1
2	encontrada	gato	Gato blanco encontrado en la calle San Martín.	2025-09-20 10:15:00	Calle San Martín 123	https://example.com/foto2.jpg	validado	2
3	callejero	perro	Cachorro abandonado cerca de la terminal de ó...	2025-09-25 18:45:00	Terminal de ómnibus, Ciudad	https://example.com/foto3.jpg	difundido	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
98      -- Ver todos los usuarios
99 •    SELECT * FROM Usuario;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_usuario	nombre	email	contrasena	rol	estado
1	Ana Pérez	ana.perez@gmail.com	contra1	usuario	activo
2	Juan López	juan.lopez@gmail.com	contra2	administrador	activo
3	María Gómez	maria.gomez@gmail.com	contra3	voluntario	bloqueado
NULL	NULL	NULL	NULL	NULL	NULL

Borrado de registros

```
124 •   DELETE FROM Usuario;
```

Result Grid | Filter Rows: | Edit:

id_usuario	nombre	email	contrasena	rol	estado
NULL	NULL	NULL	NULL	NULL	NULL

```
116 •   DELETE FROM AuditoriaCambios;
```

Result Grid | Filter Rows: | Edit:

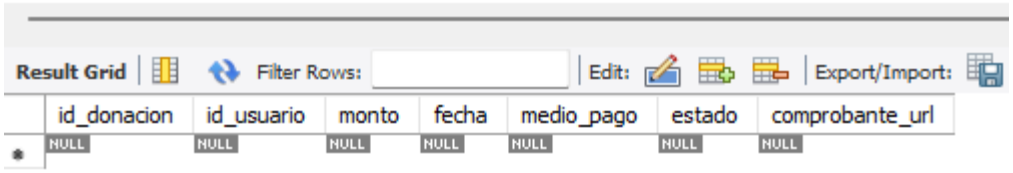
id_auditoria	id_reporte	id_usuario	fecha	cambio_realizado
NULL	NULL	NULL	NULL	NULL

```
118 •   DELETE FROM Difusion;
```

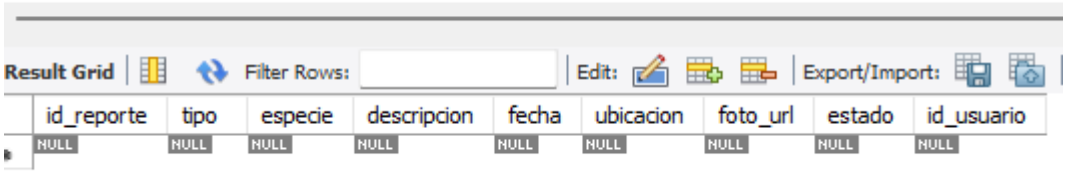
Result Grid | Filter Rows: | Edit:

id_difusion	id_reporte	fecha_publicacion	red_social	estado
NULL	NULL	NULL	NULL	NULL

```
120 •   DELETE FROM Donacion;
```




```
122 •   DELETE FROM Reporte;
```



Definiciones de comunicación

El sistema debe garantizar la correcta interacción entre sus distintos componentes, incluyendo la interfaz de usuario, el servidor de aplicaciones, la base de datos y servicios externos como redes sociales y pasarelas de pago (futuras).

Comunicación interna

Usuario ↔ Aplicación Java (Frontend ↔ Backend):

- El usuario podrá interactuar desde PC, notebook o celular mediante la interfaz gráfica (Swing/JavaFX) o consola.
- Los datos de reportes, donaciones y consultas se envían al backend utilizando HTTPS para proteger la información en tránsito.

Backend ↔ Base de datos MySQL:

- El backend gestiona la lógica de negocio (usuarios, reportes, donaciones, difusión) y realiza operaciones CRUD en MySQL mediante JDBC sobre TCP/IP.
- Se garantiza la integridad de los datos mediante transacciones ACID y validaciones del sistema.

Comunicación con servicios externos

Difusión en redes sociales:

- Se emplean las APIs de Facebook e Instagram mediante HTTPS y formato JSON.
- La autenticación y autorización se realiza mediante OAuth 2.0, garantizando seguridad y control de acceso.

Pasarelas de pago (requerimiento futuro):

- Se prevé la integración con APIs externas de pago (MercadoPago, tarjetas de crédito) mediante REST o SOAP sobre HTTPS.
- La integración futura deberá cumplir con estándares de seguridad PCI-DSS para la transmisión de datos financieros.

Infraestructura de red y física

- Red local (LAN) para desarrollo y pruebas, con routers y switches estándar.

- Red pública (Internet) para la comunicación con APIs externas y acceso remoto de usuarios.
- Servidor de aplicaciones con Java/JVM (XAMPP/Apache) y servidor MySQL, asegurando conectividad TCP/IP estable.
- Firewall y cifrado TLS/SSL para proteger las comunicaciones.

Control de enlace de datos

- Transporte confiable mediante TCP/IP.
- Validación de integridad en la capa de aplicación para formularios y pagos.
- Disponibilidad y respaldo mediante backups periódicos de la base de datos y monitoreo de APIs externas.

Conclusión

La segunda etapa del proyecto reafirma la relevancia del sistema informático desarrollado para la gestión y difusión de mascotas perdidas y animales callejeros en la ciudad de Córdoba, consolidando el avance desde la fase de análisis hacia el diseño, la implementación y las pruebas del prototipo. A través de la aplicación continua del Proceso Unificado de Desarrollo (PUD), se logró garantizar la trazabilidad entre los diferentes artefactos del sistema —desde los casos de uso y diagramas UML hasta la base de datos y las pruebas—, asegurando coherencia y calidad en cada etapa del ciclo de desarrollo.

El diseño del modelo de datos relacional permitió representar de forma eficiente las entidades y relaciones más relevantes (usuarios, reportes, adopciones, donaciones y refugios), aplicando principios de normalización y garantizando la integridad referencial en MySQL.

Asimismo, se definieron los protocolos y mecanismos de comunicación necesarios para asegurar la correcta interacción entre los componentes del sistema y los servicios externos, contemplando la escalabilidad futura hacia integraciones con redes sociales y pasarelas de pago.

En síntesis, esta entrega refleja el paso de una propuesta conceptual a un sistema funcional y operativo, evidenciando cómo un enfoque metodológico estructurado y una arquitectura bien definida pueden transformar una necesidad social en una solución tecnológica concreta, sostenible y de impacto comunitario.

Referencias

- Aplicación MySQL Workbench.
- Caso ECOViento, material de Canvas.
- Clase sincrónica, dictada por el profesor Pablo Alejandro Virgolini.
- Lucidchart para elaboración de los distintos diagramas. https://www.lucidchart.com
- Mi repositorio GitHub https://github.com/Florencia-otero1/Seminario-Siglo.git
- Módulo 2 “Proceso unificado de desarrollo”, material de Canvas.