PetLife

Documento de diseño

Proyecto Final

Cervantes Coss Anibal Alejandro
Cruz Trejo Said Carbot
López Hernández Julián
Mora Hernández Ángel Fernando
Análisis y Diseño de Sistemas

13 de enero de 2026
(Entrega Final)

Índice general

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Índice de figuras

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Índice de tablas

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 1

Introducción

Este documento presenta el diseño del sistema **PetLife**. Su contenido se basa en el análisis de requerimientos realizado previamente y funciona como una guía técnica para las etapas de construcción, pruebas y puesta en marcha del sistema.

PetLife es una plataforma para la administración integral de una clínica veterinaria. El diseño propuesto utiliza una arquitectura que combina una aplicación web tradicional con un módulo de Inteligencia Artificial para apoyar el diagnóstico clínico.

En las siguientes secciones se detallan las propiedades del software, las reglas de diseño, la arquitectura del sistema, y los modelos necesarios (estáticos y dinámicos) para entender cómo funciona la solución por dentro.

1.1. Propósito del documento

El propósito principal de este documento es definir claramente cómo está diseñado el sistema PetLife. Sirve como referencia para que el equipo de desarrollo entienda la estructura del proyecto y cómo deben interactuar sus componentes.

Este documento permite:

- Entender la arquitectura general y la división entre la web (PHP) y la IA (Python).
- Conocer las directivas y reglas que guían el diseño.
- Ver cómo el diseño resuelve los requisitos de calidad (seguridad, velocidad, etc.).
- Tener una base sólida para escribir el código y configurar la base de datos.

1.2. Alcance del diseño

El diseño del sistema PetLife cubre los siguientes puntos técnicos:

- **Propiedades de Software:** Descripción de los requisitos no funcionales y cómo se solucionan técnicamente.

- **Directivas de Diseño:** Lista de normas y enlaces a los estándares usados.
- **Modelo Arquitectónico:** Diagrama de despliegue que muestra los servidores y nodos del sistema.
- **Modelo Estático:** Diagramas de paquetes y clases que explican la organización del código.
- **Modelo Dinámico:** Diagramas de secuencia que muestran el paso a paso de las funciones principales.
- **Modelo de Persistencia:** Lista de los comandos SQL necesarios para manejar los datos.

1.3. Descripción general del sistema

PetLife es un sistema diseñado para facilitar el trabajo diario en una veterinaria. Permite registrar dueños, gestionar pacientes, agendar citas y guardar el historial médico.

Desde el punto de vista técnico, el sistema funciona con una arquitectura cliente-servidor dividida en dos partes:

1. **Módulo Web:** Es la parte principal donde interactúan los usuarios (veterinarios y administradores). Está hecha con PHP y gestiona la base de datos.
2. **Servicio de IA:** Es un componente separado hecho en Python. Se encarga de procesar datos clínicos para hacer predicciones (como la esperanza de vida) sin afectar la velocidad del sitio web.

Esta división permite que el sistema sea más ordenado y fácil de mantener en el futuro.

1.4. Organización del documento

El documento sigue el siguiente orden para explicar el sistema paso a paso:

- **Capítulo 2 (Propiedades de Software):** Explica qué características de calidad debe tener el sistema (como seguridad y rapidez) y cómo se logran.
- **Capítulo 3 (Directivas de Diseño):** Enumera las reglas y tecnologías que se deben respetar.
- **Capítulo 4 (Modelo Arquitectónico):** Muestra un mapa general de cómo se conectan los servidores y componentes.
- **Capítulo 5 (Modelo Estático):** Detalla cómo están organizadas las clases y archivos del código.
- **Capítulo 6 (Modelo Dinámico):** Usa diagramas para explicar qué pasa internamente cuando un usuario usa el sistema.
- **Capítulo 7 (Modelo de Persistencia):** Muestra las consultas SQL que el sistema realiza a la base de datos.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 2

Propiedades de Software

En el presente capítulo se establecen los Requerimientos No Funcionales (RNF) que rigen el comportamiento y la calidad del sistema **PetLife**. A partir de la investigación realizada y basándose en el modelo de calidad de la norma **ISO/IEC 25010** y la metodología *Architecture-Centric Design Method* (ACDM), se han definido las tácticas arquitectónicas necesarias. Dado que el proyecto implementa una arquitectura híbrida (interfaz web en PHP y módulo de Inteligencia Artificial en Python), el diseño contempla estrategias específicas para la integración y estabilidad de ambos entornos.

2.1. Descripción de los requerimientos no funcionales

A continuación, se detallan las propiedades de software identificadas como críticas para el proyecto, organizadas por atributo de calidad.

2.1.1. Seguridad

La seguridad del sistema se orienta a la protección de la integridad de los datos clínicos y la privacidad de la información de los usuarios. El diseño incorpora mecanismos de defensa en profundidad tanto en la capa de aplicación web como en la interfaz de programación (API) del modelo predictivo.

2.1.2. Rendimiento

El rendimiento constituye un factor crítico para la experiencia del usuario, especialmente en el procesamiento de imágenes y la ejecución del modelo de Inteligencia Artificial.

2.1.3. Disponibilidad

La disponibilidad garantiza la continuidad operativa del sistema ante fallos parciales de servicios o infraestructura.

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-SEG-01	Uso de sentencias preparadas en PHP para prevenir inyecciones SQL.
RNF-SEG-02	Hashing seguro de contraseñas mediante `password_hash()`.
RNF-SEG-03	Validación de sesiones activas antes de permitir acceso a módulos protegidos.
RNF-SEG-04	Validación de tipo MIME y renombramiento seguro de archivos de imagen.
RNF-SEG-05	Escapado de caracteres HTML para prevenir ataques XSS.
RNF-SEG-06	Validación de datos de entrada en la API mediante esquemas Pydantic.

Tabla 2.1: Propiedades de Seguridad y tácticas aplicadas

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-PERF-01	Carga única del modelo de IA en memoria al iniciar el servidor.
RNF-PERF-02	Compresión automática de imágenes a formato WebP.
RNF-PERF-03	Comunicación asíncrona mediante AJAX sin recarga de página.
RNF-PERF-04	Indexación de campos críticos en la base de datos MySQL.
RNF-PERF-05	Minificación y caché de recursos estáticos CSS y JavaScript.

Tabla 2.2: Estrategias de Rendimiento

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-DISP-01	Manejo de excepciones ante fallos del servicio de IA.
RNF-DISP-02	Uso de transacciones SQL para preservar integridad de datos.
RNF-DISP-03	Endpoint de verificación de estado del servicio de IA.
RNF-DISP-04	Copias de seguridad periódicas de archivos multimedia.
RNF-DISP-05	Centralización de configuraciones y credenciales.

Tabla 2.3: Estrategias de Disponibilidad

2.1.4. Escalabilidad

La escalabilidad permite al sistema crecer en funcionalidades y volumen de datos sin afectar su estabilidad.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-ESC-01	Desacoplamiento entre frontend PHP y API de IA en Python.
RNF-ESC-02	Arquitectura modular para nuevos tipos de predicción.
RNF-ESC-03	Organización jerárquica del almacenamiento de imágenes.
RNF-ESC-04	Base de datos relacional normalizada para múltiples usuarios.
RNF-ESC-05	Separación entre lógica de negocio y presentación.

Tabla 2.4: Estrategias de Escalabilidad

2.1.5. Usabilidad

La usabilidad busca una interacción clara e intuitiva para veterinarios y propietarios de mascotas.

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-USA-01	Diseño visual consistente mediante hojas de estilo comunes.
RNF-USA-02	Retroalimentación inmediata mediante validaciones asíncronas.
RNF-USA-03	Diseño responsivo con media queries CSS.
RNF-USA-04	Representación gráfica clara de resultados del modelo de IA.
RNF-USA-05	Navegación persistente entre módulos principales.

Tabla 2.5: Estrategias de Usabilidad

2.1.6. Mantenibilidad

La mantenibilidad permite corregir, adaptar y evolucionar el sistema con bajo costo técnico.

Requerimiento (RNF)	Solución en el Diseño (Táctica)
RNF-MANT-01	Uso de tipado y documentación automática en la API.
RNF-MANT-02	Gestión explícita de dependencias en archivos de configuración.
RNF-MANT-03	Centralización de la lógica de acceso a datos.
RNF-MANT-04	Organización semántica de la estructura de directorios.
RNF-MANT-05	Implementación de mecanismos de logging detallado.

Tabla 2.6: Estrategias de Mantenibilidad

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Bibliografía

[1] International Organization for Standardization. (2011). *ISO/IEC 25010: Systems and software quality models*.

[2] Lattanze, A. J. (2009). *Architecting Software Intensive Systems*. Auerbach Publications.

[3] Martin, R. C. (2017). *Clean Architecture*. Prentice Hall.

[4] Ramírez, S. (2024). *FastAPI Documentation*.

[5] The PHP Group. (2024). *PHP Manual: Security and Database Access*.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 3

Directivas de Diseño

Este capítulo establece los principios, restricciones y guías que orientan el desarrollo del sistema PetLife. Estas directivas aseguran la coherencia técnica entre los módulos desarrollados por los distintos integrantes del equipo y garantizan que el software cumpla con los atributos de calidad definidos previamente.

3.1. Introducción a las directivas de diseño

El diseño del sistema PetLife se rige por la simplicidad y la separación de responsabilidades. Dado que el sistema integra tecnologías heterogéneas (PHP y Python), es fundamental establecer reglas claras sobre cómo interactúan estos componentes para evitar un sistema frágil o difícil de mantener. Las siguientes directivas son de cumplimiento obligatorio para la implementación de nuevos módulos.

3.2. Directivas arquitectónicas

Estas directrices definen la estructura de alto nivel del sistema y la organización de sus componentes físicos y lógicos.

3.2.1. Separación por capas

El sistema debe mantener una estricta separación entre la lógica de presentación, la lógica de negocio y el acceso a datos.

- **Capa de Presentación:** Archivos HTML/PHP que solo deben contener lógica de visualización (bucles de renderizado, condicionales de interfaz). No deben realizar consultas SQL directas.
- **Capa de Negocio:** Scripts que procesan la información (ej. validaciones, llamadas a la API de Python).
- **Capa de Persistencia:** La comunicación con la base de datos debe centralizarse en archivos de configuración o clases específicas, evitando credenciales dispersas en el código.

3.2.2. Bajo acoplamiento

Para minimizar la dependencia entre el sistema web y el módulo de Inteligencia Artificial, la comunicación debe ser exclusivamente a través de servicios HTTP (API REST). El código PHP no debe depender de librerías internas de Python, ni viceversa. Si el servidor de IA se detiene, el sitio web principal debe seguir operando en sus funciones administrativas básicas.

3.2.3. Alta cohesión

Cada módulo o archivo debe tener una responsabilidad única y bien definida.

- Los archivos dentro de `petlife_api/routers/` deben encargarse únicamente de recibir peticiones y llamar al modelo, no de gestionar usuarios de la web.
- Los estilos visuales deben permanecer en archivos CSS externos y nunca mezclados en el código PHP/HTML (inline styles).

3.2.4. Modularidad

El sistema se divide en módulos funcionales independientes (Gestión de Usuarios, Gestión de Mascotas, Servicio de IA). Cualquier nueva funcionalidad debe clasificarse dentro de uno de estos módulos o justificar la creación de uno nuevo, manteniendo la estructura de directorios establecida en el diseño.

3.3. Directivas de diseño orientado a objetos

Aunque parte del sistema utiliza programación estructurada (scripts funcionales), se aplican principios de orientación a objetos para la gestión de conexiones y el manejo de datos complejos.

3.3.1. Encapsulamiento

El acceso directo a las propiedades de las clases o estructuras de datos críticas (como la conexión a la base de datos) debe estar restringido. Se deben utilizar métodos de acceso (getters/setters) o funciones intermediarias para modificar el estado de los objetos, protegiendo así la integridad de los datos.

3.3.2. Abstracción

Se deben ocultar los detalles complejos de implementación. Por ejemplo, el módulo web debe ”saber”que puede pedir una predicción de vida a la IA, pero no necesita conocer los detalles internos del modelo *Joblib* ni el preprocesamiento matemático que realiza Python.

3.3.3. Uso de interfaces claras

La comunicación entre el cliente (JavaScript/jQuery) y el servidor (PHP/Python) debe realizarse a través de interfaces definidas (Endpoints) que acepten y retornen datos en formato estándar JSON. Esto permite que el frontend cambie visualmente sin romper la lógica del backend.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

3.4. Directivas de codificación y estándares

Siguiendo las instrucciones del proyecto, no se transcriben los manuales de estilo en este documento. En su lugar, se adoptan los estándares internacionales de la industria para los lenguajes utilizados en PetLife. El código fuente debe adherirse a las siguientes especificaciones:

3.4.1. Estándares de codificación

Para el módulo Web (PHP): Se adopta el estándar **PSR-12: Extended Coding Style Guide**. Este define el uso de sangrías, declaración de clases, visibilidad de métodos y estructuras de control.

- **Referencia:** https://www.php-fig.org/psr/psr-12/

Para el módulo de IA (Python): Se adopta la guía de estilo **PEP 8 – Style Guide for Python Code**. Regula la indentación, el nombrado de variables (snake_case) y la longitud de líneas.

- **Referencia:** https://peps.python.org/pep-0008/

Para el Frontend (HTML/CSS): Se siguen las guías de estilo de Google para HTML y CSS, priorizando la semántica de las etiquetas y el orden de los atributos.

- **Referencia:** https://google.github.io/styleguide/htmlcssguide.html

3.4.2. Referencia a normas y guías externas

Adicionalmente, el diseño de la base de datos se rige por las reglas de normalización estándar (hasta la Tercera Forma Normal - 3FN) y la nomenclatura de tablas en plural y minúsculas, conforme a las buenas prácticas de MySQL.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 4

Modelo Arquitectónico

Este capítulo presenta la estructura fundamental del sistema **PetLife**. Se detallan los componentes de software, su distribución en nodos de hardware y la organización lógica que permite satisfacer los requisitos funcionales y de calidad establecidos anteriormente.

4.1. Descripción de la arquitectura del sistema

La arquitectura del sistema PetLife se ha diseñado como una solución web integral que combina la gestión administrativa tradicional con capacidades de análisis predictivo. El sistema no es un monolito indivisible, sino que opera mediante la orquestación de dos entornos de ejecución distintos que coexisten en el servidor de aplicaciones:

1. **Entorno Web (PHP):** Responsable de la interfaz de usuario, la lógica de negocio transaccional (citas, registros) y la seguridad de acceso.
2. **Entorno de Inteligencia Artificial (Python):** Responsable de la carga y ejecución del modelo predictivo (`.joblib`), expuesto como un servicio interno para ser consumido por el entorno web.

Esta división permite que el sistema aproveche la robustez de PHP para la web y la potencia de las librerías de ciencia de datos de Python, comunicándose entre sí de manera transparente para el usuario final.

4.2. Estilo arquitectónico

El diseño adopta principalmente el estilo **Cliente-Servidor** en su interacción con los usuarios, complementado internamente por un enfoque de **Arquitectura en Capas** y principios de **Microservicios**.

- **Patrón Cliente-Servidor:** Los usuarios interactúan a través de un navegador web (Cliente ligero) que realiza peticiones HTTP a un servidor centralizado donde reside la lógica.
- **Arquitectura en Capas (Layered):** El código se organiza en capas de Presentación (HTML/CSS), Lógica de Negocio (PHP/Python) y Persistencia (MySQL), asegurando la separación de responsabilidades.

- **Desacoplamiento de Servicios:** El módulo de IA funciona de manera autónoma, similar a un microservicio. Esto significa que la caída del servicio de predicción no detiene la operatividad del resto de la plataforma administrativa.

4.3. Diagrama de despliegue

El siguiente diagrama UML ilustra la disposición física de los nodos del sistema y los protocolos de comunicación establecidos entre los componentes de software.

Figura 4.1: Diagrama de despliegue del sistema PetLife, mostrando la integración híbrida entre PHP y Python.

4.4. Explicación del diagrama de despliegue

El diagrama presentado en la Figura 4.1 se compone de tres nodos principales, cuyas responsabilidades se describen a continuación:

4.4.1. Nodo Cliente (Client Tier)

Representa el dispositivo del usuario final (computadora de escritorio o tableta en el consultorio). En este nodo se ejecuta el navegador web que interpreta la interfaz construida con HTML5, CSS3 y la librería jQuery. Su función es capturar los datos y presentarlos, delegando todo el procesamiento pesado al servidor.

4.4.2. Nodo Servidor de Aplicaciones (Application Tier)

Es el núcleo del sistema y aloja la lógica de negocio. Este nodo presenta una particularidad en su configuración interna, ya que ejecuta dos procesos servidores:

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

- **Servidor Web (PHP):** Recibe las peticiones del cliente (puerto 80/443). Procesa la autenticación, gestiona las sesiones y sirve los archivos estáticos. Actúa como el orquestador principal.
- **Servicio de IA (Python):** Se ejecuta en un puerto interno (ej. 8000). Mantiene el modelo `petlife_model.joblib` cargado en memoria RAM. Recibe datos en formato JSON desde el componente PHP, realiza la inferencia matemática y devuelve el resultado.

La comunicación entre PHP y Python se realiza mediante peticiones HTTP locales (REST), garantizando la interoperabilidad.

4.4.3. Nodo de Datos (Data Tier)

Aloja el Sistema Gestor de Base de Datos (SGBD) MySQL. Este nodo es responsable de almacenar de forma persistente la información de usuarios, mascotas, citas y las rutas de los archivos multimedia. Solo el componente PHP tiene permisos para establecer conexiones TCP/IP con este nodo, protegiendo los datos de accesos externos directos.

4.5. Requerimientos no funcionales resueltos por la arquitectura

La selección de esta arquitectura híbrida y estratificada responde directamente a las necesidades de calidad planteadas en el capítulo de Propiedades de Software:

Resolución de la Escalabilidad: La separación del motor de IA (Python) del gestor de contenidos (PHP) permite escalar cada componente por separado. Si la demanda de predicciones aumenta, el servicio Python puede migrarse a un servidor con mayor capacidad de cómputo (GPU) sin afectar al servidor web que gestiona las citas.

Resolución del Rendimiento: Al mantener el servicio de IA como un proceso persistente en el servidor, se evita la costosa tarea de cargar el modelo `.joblib` desde el disco en cada petición. El modelo reside en memoria (Singleton), permitiendo tiempos de respuesta inmediatos para el usuario.

Resolución de la Seguridad: El diseño de despliegue aísla la base de datos del cliente. El navegador nunca tiene acceso directo al nodo de datos; todas las transacciones son mediadas y sanitizadas por la capa de lógica en el servidor de aplicaciones, reduciendo la superficie de ataque.

Resolución de la Mantenibilidad: La arquitectura respeta la estructura de directorios del proyecto. Los equipos de desarrollo pueden trabajar en paralelo: mejoras en la interfaz web (carpeta `recursosWeb`) no interfieren con la optimización de los algoritmos de predicción (carpeta `petlife_api`), gracias al bajo acoplamiento entre módulos.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 5

Modelo Estático

En este capítulo se describe la estructura interna del sistema **PetLife**. A diferencia del modelo arquitectónico, que se enfoca en nodos y servidores, el modelo estático detalla cómo está organizado el código fuente, la agrupación de archivos en paquetes lógicos y las relaciones entre las clases y entidades que conforman la solución.

5.1. Descripción general del modelo estático

El diseño estático de PetLife refleja la naturaleza híbrida del proyecto. El código se encuentra particionado en dos grandes subsistemas lógicos que coinciden con la estructura de directorios del proyecto:

1. **Subsistema Web (PHP):** Encargado de la gestión de usuarios, mascotas y la interfaz gráfica. Sigue una estructura modular donde cada script o archivo cumple una función específica (Vista o Controlador).
2. **Subsistema de IA (Python):** Organizado bajo patrones de diseño de microservicios, utilizando routers para los puntos de entrada y clases para la lógica de predicción.

Esta separación permite que el mantenimiento del código sea ordenado, evitando la mezcla de lenguajes y responsabilidades en un mismo directorio.

5.2. Diagrama de paquetes

El diagrama de paquetes muestra la organización de alto nivel del código fuente, agrupando los elementos en subsistemas lógicos según su funcionalidad.

5.3. Explicación del diagrama de paquetes

La organización mostrada en la Figura 5.1 corresponde directamente con la estructura de directorios del proyecto, facilitando la navegación para los desarrolladores:

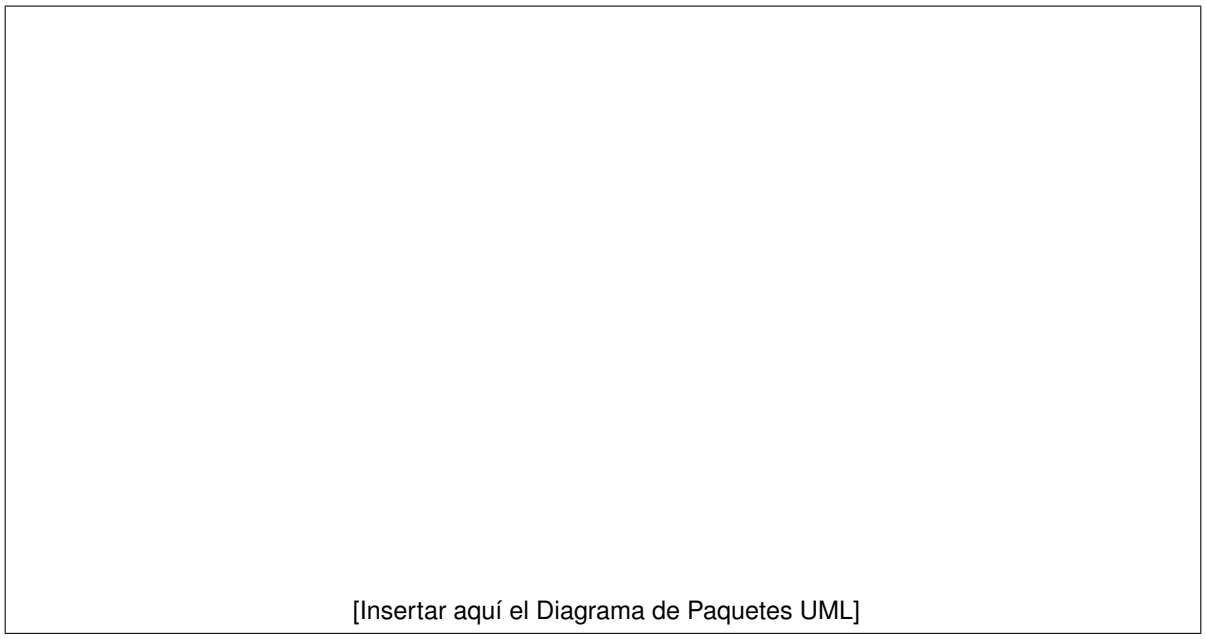

Figura 5.1: Diagrama de paquetes del sistema, reflejando la estructura de directorios y dependencias.

5.3.1. Paquete de Interfaz (Frontend)

Agrupa todos los recursos estáticos encargados de la presentación visual. Corresponde a los directorios `estilo/` (CSS) y `js/` (jQuery). Este paquete no contiene lógica de negocio, solo reglas de visualización y comportamiento del lado del cliente.

5.3.2. Paquete de Lógica Web (Backend PHP)

Contiene los scripts principales del sistema alojados en `recursosWeb/`. Este paquete actúa como el controlador del sistema, recibiendo las peticiones del usuario, validando los datos y coordinando las respuestas. Incluye archivos clave como `user.php` y `mis_mascotas.php`.

5.3.3. Paquete de Servicios de IA (API Python)

Encapsula toda la lógica de Inteligencia Artificial dentro del directorio `petlife_api/`. Este paquete es autónomo y contiene sus propios sub-paquetes:

- **Routers:** Define los puntos de entrada (endpoints) de la API.
- **Models:** Contiene los archivos binarios del modelo predictivo (`.joblib`).

5.3.4. Paquete de Datos (Persistencia)

Representa la capa de almacenamiento. Incluye los scripts de configuración de base de datos (`config.php`, `database.py`) y la estructura de directorios `uploads/` donde se almacenan las imágenes de las mascotas.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

5.4. Diagramas de clases

Los diagramas de clases detallan la estructura de las entidades del sistema, sus atributos y métodos, así como las relaciones entre ellas. Dado que el sistema utiliza una base de datos relacional y un servicio de objetos en Python, se representan las clases principales de ambos contextos.

Figura 5.2: Diagrama de clases principales del dominio de PetLife.

5.5. Explicación de los diagramas de clases

El modelo de clases representa las entidades fundamentales del negocio y cómo interactúan para cumplir los requisitos funcionales. A continuación se describen las clases más relevantes ilustradas en la Figura 5.2:

5.5.1. Clase Usuario

Representa a los actores que interactúan con el sistema (Dueños o Veterinarios).

- **Responsabilidad:** Gestionar la autenticación y los datos personales.
- **Atributos clave:** Identificador único, credenciales cifradas y datos de contacto.
- **Relaciones:** Tiene una relación de uno a muchos $(1..*)$ con la clase **Mascota**, indicando que un usuario puede ser propietario de múltiples animales.

5.5.2. Clase Mascota

Es la entidad central del sistema, sobre la cual giran la mayoría de las operaciones clínicas.

- **Responsabilidad:** Almacenar la información biológica del animal (raza, edad, peso) y referencias a sus archivos multimedia (fotos).
- **Métodos principales:** Incluye lógica para registrar nuevos ingresos y solicitar análisis al módulo de IA.

5.5.3. Clase ServicioPredictivo (Módulo IA)

Esta clase es una abstracción del componente de Python encargado de la Inteligencia Artificial.

- **Responsabilidad:** Cargar el modelo `petlife_model.joblib` en memoria y exponer el método de inferencia.
- **Interacción:** No persiste datos en la base de datos directamente; su función es procesar los atributos de la clase **Mascota** y retornar un valor calculado (ej. esperanza de vida estimada), que posteriormente es mostrado en la interfaz.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 6

Modelo Dinámico

En este capítulo se modela el comportamiento del sistema **PetLife** a través del tiempo. Mientras que el modelo estático define la estructura, el modelo dinámico describe cómo interactúan los objetos y componentes del sistema mediante el intercambio de mensajes para ejecutar los casos de uso definidos en el análisis.

6.1. Descripción general del modelo dinámico

La dinámica del sistema sigue un flujo secuencial característico de las aplicaciones web transaccionales, con la particularidad de la integración asíncrona para el módulo de Inteligencia Artificial.

Las interacciones se clasifican en dos patrones principales:

1. **Interacción Síncrona (Gestión Administrativa):** El usuario solicita una operación (ej. registrar mascota), el servidor PHP procesa la lógica, interactúa con la base de datos MySQL y devuelve una respuesta inmediata visualizada en el navegador.

2. **Interacción de Servicio (Predicción IA):** El servidor web actúa como cliente del servicio de IA (Python). Se envía una solicitud de cálculo, el servicio procesa los datos con el modelo `.joblib` y retorna un resultado JSON que el sistema web interpreta y presenta al usuario.

6.2. Diagramas de secuencia

A continuación, se presentan los diagramas de secuencia correspondientes a los procesos críticos del sistema, detallando el ciclo de vida de los objetos y el flujo de control.

6.2.1. Diagrama de secuencia: Registro de Usuario

Este diagrama ilustra el proceso mediante el cual un nuevo veterinario o dueño se da de alta en la plataforma, asegurando la validación de datos y el almacenamiento seguro de credenciales.

Figura 6.1: Secuencia de operaciones para el registro de un nuevo usuario en el sistema.

Explicación: El flujo inicia cuando el actor envía el formulario. El script `registrar_usuario.php` valida la integridad de los campos, aplica el algoritmo de hashing a la contraseña para cumplir con el RNF de Seguridad, e inserta el registro en la base de datos. Finalmente, confirma la operación al usuario.

6.2.2. Diagrama de secuencia: Predicción de Salud con IA

Este diagrama representa la funcionalidad más compleja del sistema: la comunicación entre el entorno web y el motor de inteligencia artificial para obtener una predicción clínica.

Explicación: Se observa el desacoplamiento entre capas. El usuario interactúa con la web, pero el procesamiento matemático ocurre externamente. El archivo `usar_modelo.php` actúa como puente, formateando los datos de la mascota y enviándolos a la API de Python. La respuesta se recibe asíncronamente y se actualiza la interfaz sin recargar la página.

6.2.3. Diagramas de secuencia adicionales: Gestión de Mascotas

El siguiente diagrama detalla el proceso de alta de una mascota, incluyendo la gestión de archivos multimedia (foto de perfil).

Explicación: Este proceso involucra tanto a la base de datos como al sistema de archivos del servidor. Es crucial notar que en la base de datos solo se guarda la referencia (ruta relativa) de la imagen, mientras que el archivo físico se deposita en el directorio `uploads/`, optimizando así el almacenamiento.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

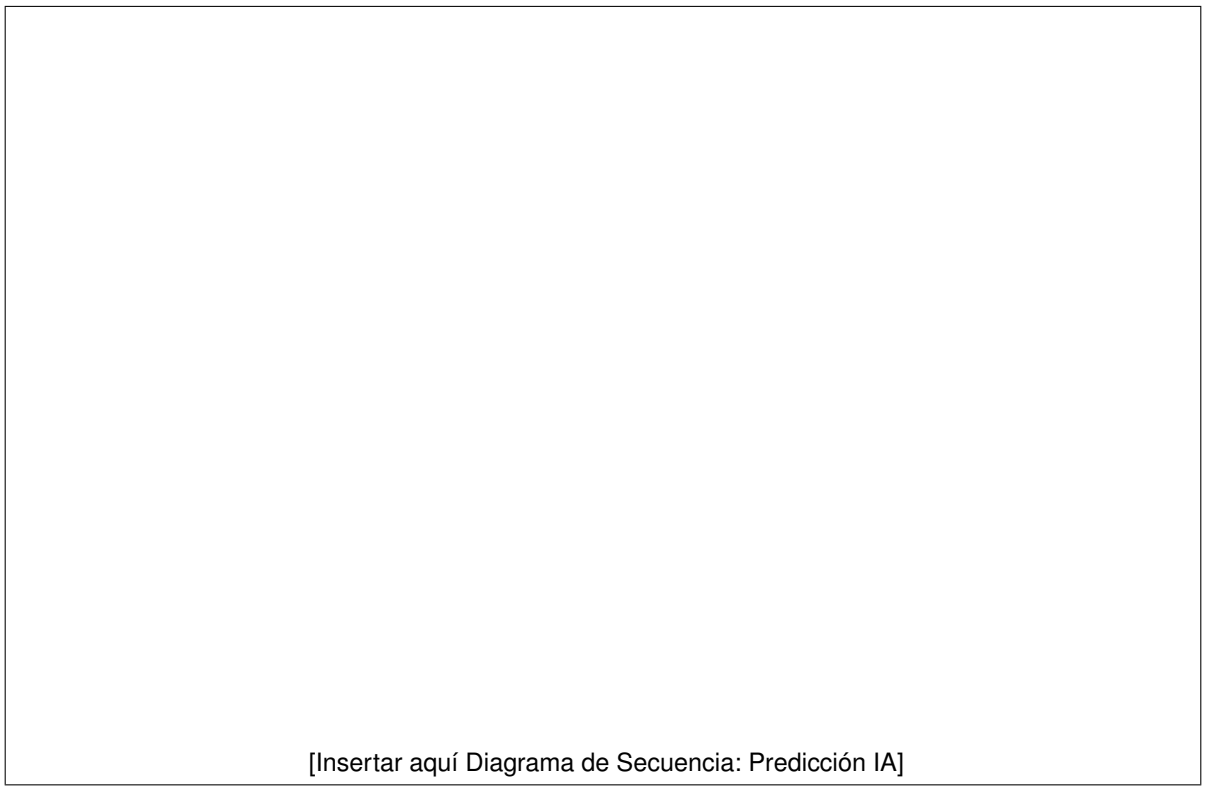

Figura 6.2: Flujo de interacción híbrido entre PHP y Python para la predicción de salud.

Figura 6.3: Secuencia de alta de mascota con gestión de almacenamiento de imágenes.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

CAPÍTULO 7

Modelo de Persistencia

Este capítulo final detalla la capa de datos del sistema **PetLife**. Se describe el esquema de almacenamiento relacional diseñado para garantizar la integridad de la información y se documentan las instrucciones SQL (Structured Query Language) que el sistema ejecuta durante los flujos descritos en el modelo dinámico.

7.1. Descripción del modelo de persistencia

El sistema implementa un modelo de persistencia relacional utilizando el Sistema Gestor de Base de Datos (SGBD) **MySQL**. La elección de esta tecnología responde a la necesidad de mantener relaciones fuertes entre las entidades (ej. Un Dueño → Muchas Mascotas) y garantizar las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) en las transacciones.

La conexión desde el módulo web (PHP) se realiza mediante la interfaz **PDO (PHP Data Objects)**, lo que permite abstraer el acceso a datos y utilizar sentencias preparadas para mitigar riesgos de seguridad. Cabe destacar que, por eficiencia, los archivos binarios (imágenes de mascotas) no se almacenan en la base de datos; en su lugar, se persiste únicamente la ruta relativa del archivo (`VARCHAR`), mientras que el recurso físico reside en el sistema de archivos del servidor.

7.2. Diseño de la base de datos

El esquema de la base de datos se encuentra normalizado hasta la Tercera Forma Normal (3FN) para evitar redundancias. A continuación se presenta el Modelo Entidad-Relación (MER) que estructura las tablas del sistema.

7.3. Comandos SQL utilizados

A continuación, se listan las sentencias SQL fundamentales que corresponden a las operaciones identificadas en los Diagramas de Secuencia del Capítulo 6. Se utiliza la notación de parámetros (‘¿) para indicar que se trata de sentencias preparadas seguras.

Figura 7.1: Diagrama de la base de datos ‘usuarios.sql‘ mostrando las tablas principales y sus relaciones.

7.3.1. Comandos SELECT (Lectura)

Estas instrucciones se ejecutan cuando el usuario navega por el sistema o cuando el servicio de IA requiere datos para procesar una predicción.

- **Inicio de Sesión:** Recuperación de credenciales para validar el acceso.

```
SELECT id, password, rol FROM usuarios WHERE email = ? LIMIT 1;
```

- **Listado de Mascotas:** Obtención de todas las mascotas pertenecientes al usuario en sesión (Visualización en `mis_mascotas.php`).

```
SELECT id, nombre, raza, foto_path
FROM mascotas
WHERE id_usuario = ?;
```

- **Consulta para IA:** Obtención de datos clínicos específicos para enviar al modelo predictivo Python.

```
SELECT raza, edad, peso, historial_medico
FROM mascotas
WHERE id = ?;
```


Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

7.3.2. Comandos INSERT (Creación)

Sentencias utilizadas en los procesos de registro de nuevos usuarios y alta de pacientes.

- **Registro de Usuario:** Inserción de un nuevo actor en el sistema.

```
INSERT INTO usuarios (nombre, email, password, rol, fecha_registro)
VALUES (?, ?, ?, 'usuario', NOW());
```

- **Alta de Mascota:** Registro de un nuevo paciente vinculado a su dueño. Nótese el campo para la ruta de la imagen.

```
INSERT INTO mascotas (nombre, raza, edad, peso, foto_path, id_usuario)
VALUES (?, ?, ?, ?, ?, ?);
```

7.3.3. Comandos UPDATE (Actualización)

Operaciones ejecutadas cuando el usuario modifica información existente, como corregir datos de una mascota tras una consulta.

- **Actualización de Expediente:** Modificación de datos variables como el peso o la edad.

```
UPDATE mascotas
SET edad = ?, peso = ?
WHERE id = ? AND id_usuario = ?;
```

7.3.4. Comandos DELETE (Eliminado)

Instrucciones para la eliminación de registros. Dependiendo de la configuración del sistema, esto puede ser un borrado físico o lógico.

- **Baja de Mascota:** Eliminación del registro de la base de datos.

```
DELETE FROM mascotas
WHERE id = ? AND id_usuario = ?;
```

Nota: El diseño contempla que, antes de ejecutar este comando SQL, el sistema de archivos debe eliminar la imagen física asociada en la carpeta `uploads/` para evitar archivos huérfanos.

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Referencias Bibliográficas

Calle Gato de La Francia # 314 Col. Abidalso, Benito Juarez, CDMX ☎ 559292334 Ext. 10293 ✉ banre@petlife.com

Bibliografía

[1] K. P. García Ventura, “Clínica veterinaria digital: análisis inteligente para la gestión de casos clínicos,” Tesis de Licenciatura, Facultad de Ingeniería, Universidad Nacional Autónoma de México (UNAM), Ciudad de México, 2025. [En línea]. Disponible en: https://ru.dgb.unam.mx/handle/20.500.14330/TES01000871459

[2] Instituto Nacional de Estadística y Geografía (INEGI), “Encuesta Nacional de Bienestar Autorreportado (ENBIARE) 2021: Tenencia de mascotas en México,” *Comunicado de prensa núm. 725/21*, Ciudad de México, dic. 2021. [En línea]. Disponible en: https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2021/EstSociodemo/ENBIARE_2021.pdf

[3] Facultad de Medicina Veterinaria y Zootecnia (FMVZ), *Tu clínica veterinaria: Guía para el emprendimiento y gestión*, 1a ed., Ciudad de México: Universidad Nacional Autónoma de México (UNAM), 2021. [En línea]. Disponible en: https://www.fmvz.unam.mx/fmvz/publicaciones/archivos/Clinica_Veterinaria.pdf

[4] Secretaría de Agricultura y Desarrollo Rural (SADER), “Norma Oficial Mexicana NOM-012-ZOO-1993, Especificaciones para la regulación de productos químicos, farmacéuticos, biológicos y alimenticios para uso en animales o consumo por éstos,” *Diario Oficial de la Federación*, México, 17 ene. 1995. [En línea]. Disponible en: https://www.dof.gob.mx/normasOficiales/9371/sader/sader.html