CIENTIFICA TechNoLogica
ISSN:3119-7906, Vol.1Núm.1(2026): Enero-Marzo
https://doi.org/10.17268/tech1.1.2026.03

Aplicación de Patrones en ConectaPet: Sistema Web de Adopciones y Donaciones de Perros y Gatos

Using Templates in ConectaPet: A Web-Based System for Dog and Cat Adoptions and Donations

Gamarra Saucedo Maeli S.[1], Morales Romero Franco F.[1], Núñez Chilón Dorely J.[1], Ramírez Guarniz José A[1,*].

[1] Escuela de Ingeniería Informática, Universidad Nacional de Trujillo, La Libertad, Perú.

*Autor correspondiente: g2052700521@unitru.edu.pe (J-RamirezGuarniz)

RESUMEN

ConectaPet es una plataforma web orientada a facilitar la adopción responsable de mascotas y la gestión de donaciones destinadas a su bienestar. El sistema permite a los usuarios visualizar mascotas disponibles, registrar solicitudes de adopción y realizar donaciones, mientras que los administradores pueden gestionar mascotas, usuarios y procesos internos. La plataforma fue desarrollada utilizando el patrón arquitectónico Modelo-Vista-Controlador (MVC), junto con patrones de diseño como Singleton, Factory Method y Template Method, garantizando una estructura modular, mantenible y escalable. Los resultados demuestran una mejora en la organización del código, la reutilización de componentes y la separación de responsabilidades, contribuyendo a un sistema robusto y fácilmente extensible.
Palabras clave: adopción de mascotas, patrones de diseño, MVC, aplicaciones web, gestión de donaciones.

ABSTRACT

ConectaPet is a web platform designed to facilitate responsible pet adoption and donation management for animal welfare. The system allows users to browse available pets, submit adoption requests, and make donations, while administrators manage pets, users, and internal processes. The platform was developed using the Model-View-Controller (MVC) architectural pattern, along with design patterns such as Singleton, Factory Method, and Template Method, ensuring a modular, maintainable, and scalable structure. The results show improvements in code organization, component reuse, and separation of concerns, contributing to a robust and easily extensible system.
***Keywords**: pet adoption, design patterns, MVC, web applications, donation management.*

1. INTRODUCCIÓN

En la actualidad, las plataformas web desempeñan un papel fundamental en la optimización de procesos sociales y comunitarios, permitiendo una gestión más eficiente, transparente y accesible de la información. En el ámbito del bienestar animal, diversas organizaciones han desarrollado sistemas digitales orientados a facilitar la adopción responsable de mascotas y la gestión de donaciones, demostrando el impacto positivo de la tecnología en este tipo de iniciativas sociales.

Un ejemplo representativo es **Petfinder**, una plataforma internacional que conecta refugios y organizaciones de rescate con posibles adoptantes, permitiendo la visualización de mascotas disponibles y el envío de solicitudes de adopción en línea [1]. De manera similar, **Adopt-a-Pet** ofrece un sistema web que centraliza información de refugios y promueve la adopción responsable, mejorando la visibilidad de animales en situación de abandono [2].

En el contexto latinoamericano, plataformas como **Rescate Animal Chile** y **Patitas Perú** utilizan portales web y redes digitales para difundir información sobre mascotas en adopción, gestionar campañas de donación y fortalecer la participación ciudadana en la protección animal [3]. Estos sistemas han demostrado que la digitalización contribuye a reducir los tiempos de adopción y optimizar la gestión de recursos destinados al bienestar animal.

En este contexto surge **ConectaPet**, una plataforma web orientada a centralizar y automatizar los procesos de adopción de mascotas y gestión de donaciones. El sistema permite a los usuarios visualizar mascotas disponibles, registrar solicitudes de adopción y realizar donaciones, mientras que los administradores gestionan la información, los usuarios y los estados de los procesos de manera eficiente.

Desde el enfoque técnico, la plataforma fue desarrollada utilizando el patrón arquitectónico **Modelo–Vista–Controlador (MVC)**, complementado con patrones de diseño como **Singleton**, **Factory Method** y **Template Method**. Esta combinación de patrones favorece la separación de responsabilidades, la reutilización de componentes y la escalabilidad del sistema, contribuyendo al desarrollo de una aplicación web robusta y alineada con buenas prácticas de ingeniería de software.

2. MATERIALES Y MÉTODOS

A. Herramientas y entorno de desarrollo

El sistema fue desarrollado como un **sistema web**, permitiendo su acceso desde distintos dispositivos con conexión a Internet. El entorno de desarrollo se estructuró bajo una arquitectura **Modelo–Vista–Controlador (MVC)**, facilitando la separación de responsabilidades y mejorando la mantenibilidad del código.

El desarrollo se realizó en un entorno local utilizando las siguientes herramientas:

- **Sistema Operativo:** Windows 10
- **Servidor Web:** Apache (XAMPP)
- **Lenguaje de Programación:** PHP
- **Base de Datos:** MySQL
- **Lenguajes de Interfaz:** HTML5, CSS3 y JavaScript
- **Editor de Código:** Visual Studio Code
- **Control de Dependencias:** Composer

Los principales recursos de software utilizados en el desarrollo del sistema ConectaPet fueron los siguientes:

- **PHP:** Lenguaje principal para la lógica del sistema y comunicación con la base de datos.
- **MySQL:** Sistema de gestión de bases de datos relacional para el almacenamiento de información.
- **PHPMailer:** Librería utilizada para el envío automático de correos electrónicos relacionados con confirmaciones de donaciones y notificaciones del sistema.
- **Composer:** Gestor de dependencias para la integración de librerías externas.

B. Proceso de Desarrollo del Sistema

1) Análisis de requisitos

El estudio de requisitos de ConectaPet se orientó hacia las demandas específicas de los principales actores del sistema. Los usuarios interesados en la adopción necesitan acceder a un catálogo de mascotas con información precisa y formularios que faciliten la solicitud; los donantes requieren un módulo seguro y confiable para registrar sus aportes y obtener confirmaciones; mientras que los administradores necesitan herramientas integrales para gestionar tanto mascotas como usuarios y donaciones. Asimismo, se establecieron requisitos no funcionales relacionados con la seguridad, la escalabilidad y la consistencia visual, con el propósito de garantizar que la plataforma cumpla con estándares de calidad propios de aplicaciones web enfocadas en el bienestar animal.

2) Diseño Arquitectónico del Sistema (MVC)

El sistema se diseñó siguiendo el patrón arquitectónico **MVC**, el cual divide la aplicación en tres componentes principales:

- **Modelo:** Encargado de la gestión de los datos y la lógica del negocio, representado por clases como *Mascota*, *Donación* y *Noticia*. (Fig. 1)

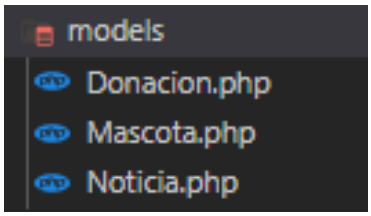

***Fig. 1**. Models*

- **Vista:** Responsable de la presentación de la información al usuario final, implementada mediante archivos PHP organizados por módulos. (Fig. 2)

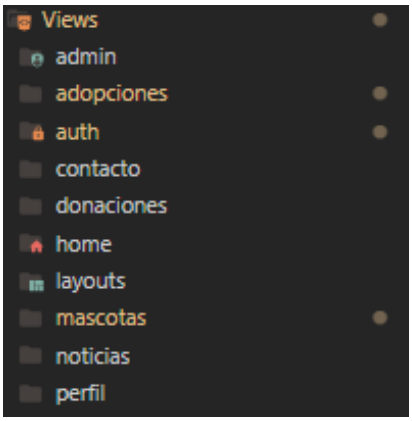

***Fig. 2.** Vista*

- **Controlador:** Gestiona la interacción entre el usuario y el sistema, procesando solicitudes y coordinando las acciones entre modelos y vistas. (Fig. 3)

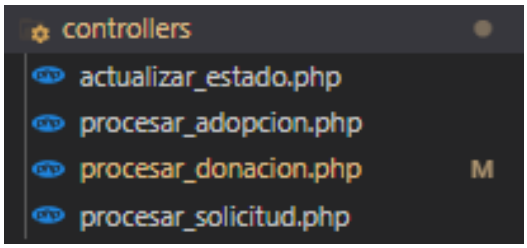

***Fig. 3**. Controllers*

Esta arquitectura permitió una organización clara del sistema, facilitando la reutilización del código y futuras ampliaciones.

3) Identificación de problemas de diseño

En la fase de diseño del sistema ConectaPet se detectaron diversos aspectos que podían afectar la calidad y la capacidad de mantenimiento de la aplicación. Los principales inconvenientes fueron:

- Repetición de código en las vistas, sobre todo en componentes compartidos como menús, encabezados y pies de página.
- Alto nivel de acoplamiento entre controladores y modelos, lo que limitaba la reutilización y aumentaba la dificultad del mantenimiento.
- Administración poco eficiente de la conexión con la base de datos, con el riesgo de generar múltiples instancias que comprometieron la consistencia de la información.
- Complejidad en la integración de la lógica de negocio con la capa de presentación, lo que

evidenció la necesidad de una arquitectura más clara que delimitara responsabilidades.

La detección de estos problemas se convirtió en un insumo clave para las siguientes etapas del proyecto, donde se plantearon soluciones concretas mediante la incorporación de patrones de diseño.

4) Patrón Singleton

El patrón de diseño Singleton es un patrón creacional que busca asegurar que una clase tenga una única instancia durante la ejecución de un sistema, proporcionando un acceso global controlado a esa instancia. Su aplicación es particularmente útil en la arquitectura de software para gestionar de forma eficiente recursos compartidos y garantizar que las operaciones centralizadas, como el acceso a servicios o datos, se realicen de manera consistente y sin duplicación de instancias [4].

En el sistema **ConectaPet**, el patrón Singleton se aplicó para gestionar la **conexión a la base de datos**, evitando la creación de múltiples instancias que podrían generar inconsistencias en la información y un uso ineficiente de los recursos del servidor.

4.1 Implementación del patrón Singleton en ConectaPet

La implementación del Singleton se realizó en la **capa Modelo**, encapsulando la lógica de conexión a la base de datos en una clase especializada (Fig. 4). Dicha clase controla internamente la creación de su propia instancia, asegurando que solo exista una conexión activa durante la ejecución del sistema.

Mediante este enfoque, todos los modelos que requieren acceso a la base de datos utilizan la misma instancia compartida (Fig. 5), sin necesidad de crear nuevas conexiones en cada operación.

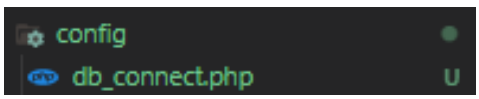

Fig. 4. *db_connect.php*

```
require_once __DIR__ . '/../../config/db_connect.php';
```

Fig. 5*. Ejemplo de uso del singleton en IndexPage.php*

4.2 Función del patrón dentro del desarrollo del proyecto

El uso del patrón Singleton en ConectaPet cumple las siguientes funciones principales:

- Garantiza una **única instancia de conexión** a la base de datos.
- Optimiza el uso de recursos del servidor, reduciendo sobrecarga.
- Asegura la consistencia de los datos durante las operaciones del sistema.
- Simplifica el acceso a la base de datos desde los distintos módulos.

4.3 Relación con la arquitectura MVC

Dentro de la arquitectura **Modelo–Vista–Controlador (MVC)**, el patrón Singleton se integra de la siguiente manera:

- **Modelo:** Utiliza la instancia única del Singleton para ejecutar consultas y operaciones sobre la base de datos.
- **Controlador:** Accede a los datos a través de los modelos, sin interactuar directamente con la conexión.
- **Vista:** No tiene acceso directo al Singleton, manteniendo la separación de responsabilidades.

Este esquema refuerza el desacoplamiento entre capas y mejora la mantenibilidad del sistema.

4.4 Beneficios del uso del Singleton en ConectaPet

La aplicación del patrón Singleton aportó beneficios significativos al desarrollo del sistema:

- Mayor **control sobre recursos compartidos**.
- Reducción de errores asociados a múltiples conexiones simultáneas.

- Código más **organizado y reutilizable**.
- Facilidad para el mantenimiento y futuras ampliaciones del sistema.

5) Patrón Factory Method

El patrón **Factory Method** es un patrón de diseño creacional cuyo objetivo principal es **centralizar la creación de objetos**, delegando esta responsabilidad a una clase especializada denominada *Factory*. De esta manera, se evita que otras partes del sistema dependan directamente de la forma en que se construyen los objetos, promoviendo un diseño más flexible y desacoplado.

En el proyecto **ConectaPet**, el patrón Factory Method se utiliza para crear instancias de las entidades principales del sistema, tales como **Mascota**, **Donación** y **Noticia**, a partir de datos provenientes de la base de datos u otras fuentes externas.

5.1) Implementación del patrón en ConectaPet

La implementación del patrón se realiza mediante clases *Factory* ubicadas en el directorio **factories/**. Cada una de estas clases es responsable de construir un objeto del modelo correspondiente, encapsulando la lógica de creación.

Por ejemplo, la clase **MascotaFactory** (Fig. 6) se encarga de crear objetos de tipo **Mascota** a partir de un arreglo asociativo que representa una fila obtenida desde la base de datos.

MascotaFactory.php M

You, 1 second ago | 1 author (You)

```php
<?php
// factories/MascotaFactory.php
require_once __DIR__ . '/../models/Mascota.php';

class MascotaFactory {

    public static function crearDesdeArray(array $data): Mascota {
        return new Mascota($data);
    }
}
```

***Fig. 6**. MascotaFactory.php*

De forma similar, la clase **DonacionFactory** crea objetos de tipo **Donacion** y lo mismo en **NoticiaFactory** con **Noticia**, garantizando que todas las instancias sean inicializadas de manera uniforme.

5.2) Función del patrón dentro del desarrollo del proyecto

El uso del patrón Factory Method en ConectaPet cumple las siguientes funciones clave:

- **Centraliza la creación de objetos**, evitando que los controladores o vistas instancien directamente los modelos.
- **Reduce el acoplamiento** entre las capas del sistema, especialmente entre el controlador y el modelo.
- **Facilita el mantenimiento**, ya que cualquier cambio en la forma de construir un objeto se realiza únicamente en la clase Factory.
- **Permite futuras extensiones**, como validaciones adicionales, valores por defecto o transformaciones de datos, sin afectar al resto del sistema.

5.3) Relación con MVC

Dentro de la arquitectura **MVC**, el patrón Factory Method actúa como un intermediario entre el **Controlador** y el **Modelo**. El controlador solicita a la Factory la creación del objeto, sin preocuparse por los detalles internos de su construcción.

Flujo simplificado:

1. El **Controlador** obtiene datos desde la base de datos.
2. La **Factory** transforma esos datos en un objeto del **Modelo**.
3. El **Modelo** es enviado a la **Vista** para su presentación.

Esto mantiene una clara separación de responsabilidades, uno de los principios fundamentales del patrón MVC.

5.4) Uso del patrón en la interfaz del sistema

Los objetos creados por las fábricas son utilizados directamente en las vistas del sistema. Por ejemplo, en la sección de **listado de mascotas**, cada tarjeta visual mostrada al usuario (Fig. 7) representa un objeto **Mascota** creado mediante **MascotaFactory**, el cual proporciona acceso a sus atributos y métodos, como el manejo seguro de la imagen (fig. 7).

Fig. 7. *Visualización sección mascotas*

6) Patrón Template Method

El patrón **Template Method** es un patrón de diseño de comportamiento que define la **estructura general de un algoritmo** en una clase base, permitiendo que las subclases implementen o personalicen ciertos pasos del proceso sin modificar la estructura principal. Este patrón promueve la reutilización de código y asegura un flujo uniforme en operaciones que comparten una lógica común.

En el proyecto **ConectaPet**, el patrón Template Method se utiliza para **definir la estructura común de las páginas web**, garantizando que todas compartan los mismos elementos base, como encabezado, menú de navegación y pie de página, mientras que el contenido específico de cada sección se personaliza según la funcionalidad requerida.

6.1) Implementación del patrón en ConectaPet

La implementación del patrón se encuentra en la clase **BaseTemplate.php**, ubicada en el directorio **layouts/**. Esta clase define el **esqueleto del proceso de renderizado** de una página web mediante un método principal que establece el orden fijo de los pasos.

6.2) Uso del patrón en las vistas

Las páginas del sistema, como **IndexPage** (Fig. 8) , **MascotasPage** o **NoticiasPage**, heredan de **BaseTemplate** y definen únicamente el contenido particular de cada sección, respetando la estructura general establecida.

```php
<?php
// pages/IndexPage.php

require_once __DIR__ . '/../layouts/BaseTemplate.php';
require_once __DIR__ . '/../../config/db_connect.php';

class IndexPage extends BaseTemplate {
    private $destacadas;
    private $campanias;
    private $actividades;
```

Fig. 8. *Uso del Método Template en IndexPage3*

6.3) Función del patrón dentro del desarrollo del proyecto

El patrón Template Method cumple funciones clave dentro del proyecto ConectaPet:

- Estandariza la estructura de las páginas web, garantizando uniformidad visual.
- Reduce la duplicación de código, ya que el encabezado y pie de página se definen una sola vez.
- Facilita el mantenimiento, permitiendo modificar la estructura general sin afectar el contenido específico.
- Mejora la escalabilidad, ya que nuevas páginas pueden crearse fácilmente heredando de la plantilla base.

6.4) Relación con la arquitectura MVC

Dentro de la arquitectura MVC, el patrón Template Method se aplica principalmente en la capa de Vista. El controlador selecciona qué vista mostrar, y la vista utiliza la plantilla base para renderizar el contenido siguiendo una estructura común.

Flujo simplificado:

1. El Controlador determina la vista a mostrar.
2. La Vista hereda de **BaseTemplate**.
3. El método **render()** ejecuta el flujo completo definido por el Template Method.

Este enfoque refuerza la separación de responsabilidades y mantiene una organización clara entre lógica y presentación.

6.5) Uso del patrón en la interfaz del sistema

El patrón Template Method se refleja directamente en la interfaz del sistema, ya que todas las páginas comparten el mismo diseño base. Por ejemplo, las

secciones de Inicio, Mascotas, Donaciones y Noticias presentan un encabezado y pie de página uniformes, variando únicamente el contenido central.

7) Integración de patrones en MVC

La arquitectura del sistema ConectaPet se fundamenta en el patrón Modelo–Vista–Controlador (MVC), el cual fue reforzado mediante la integración de patrones de diseño orientados a objetos, específicamente Singleton, Factory Method y Template Method. Esta integración permitió mejorar la organización del sistema, asegurar una correcta separación de responsabilidades y fortalecer la mantenibilidad y escalabilidad del software, tal como se plantea en estudios sobre el uso de patrones en arquitecturas MVC modernas [5].

Diversas investigaciones destacan que la combinación de patrones de diseño con MVC contribuye significativamente a la reutilización del código, la reducción del acoplamiento entre componentes y una mayor claridad estructural del sistema [6]. En este contexto, la arquitectura de ConectaPet adopta buenas prácticas ampliamente discutidas en la literatura técnica y foros especializados en ingeniería de software [7].

8) Integración del Patrón Singleton en la Arquitectura MVC

El patrón Singleton fue integrado en la capa de Modelo para la gestión de la conexión a la base de datos, garantizando la existencia de una única instancia activa durante la ejecución del sistema. Esta decisión responde a recomendaciones ampliamente documentadas sobre el uso del Singleton para el manejo eficiente de recursos compartidos en aplicaciones MVC [8].

El uso del Singleton en la conexión a la base de datos permite evitar la creación innecesaria de múltiples conexiones, reduciendo la sobrecarga del servidor y mejorando la consistencia de los datos [9]. Dentro de la arquitectura MVC, los modelos acceden a la base de datos mediante esta instancia única, mientras que los controladores interactúan exclusivamente con los modelos, sin acceder directamente al mecanismo de conexión [10].

Este enfoque se alinea con buenas prácticas de diseño que recomiendan aislar los recursos críticos en componentes controlados, evitando dependencias directas entre capas y reduciendo riesgos asociados a la concurrencia [11].

B. Integración del Patrón Factory Method en la Arquitectura MVC

El patrón Factory Method fue incorporado para centralizar la creación de objetos del modelo, tales como Mascota, Donación y Noticia. Este patrón permite desacoplar el proceso de instanciación del uso de los objetos, facilitando la evolución del sistema sin afectar a los controladores [12].

En la arquitectura MVC de ConectaPet, los controladores solicitan a las fábricas la creación de los objetos necesarios, sin conocer los detalles de su construcción interna. Esta estrategia sigue los principios del diseño orientado a objetos y coincide con las recomendaciones planteadas por expertos en patrones de diseño [13].

Además, el uso del Factory Method mejora la reutilización del código y la mantenibilidad del sistema, ya que cualquier modificación en la lógica de creación se concentra en las clases Factory, sin impactar al resto de la aplicación [14]. Esta característica resulta clave en sistemas web que requieren constantes ajustes y ampliaciones funcionales [15].

C. Integración del Patrón Template Method en la Arquitectura MVC

El patrón Template Method fue aplicado principalmente en la capa de Vista, con el objetivo de estandarizar la estructura de las páginas web del sistema. Mediante una plantilla base, se definió un flujo común de renderización que incluye encabezado, menú, contenido y pie de página, permitiendo que cada vista concreta implemente únicamente su contenido específico.

Esta integración favorece la uniformidad visual del sistema y reduce significativamente la duplicación de código, tal como se describe en material académico y notas técnicas sobre el uso del Template Method en interfaces web [16]. En el contexto de MVC, el controlador selecciona la vista correspondiente, mientras que la estructura general

de la página es gestionada por la plantilla base, manteniendo una clara separación entre lógica, control y presentación.

D. Relación Conjunta entre MVC y Patrones de Diseño

La combinación de la arquitectura MVC con los patrones de diseño aplicados permite una organización clara del sistema:

- **MVC** define la estructura general y la separación de responsabilidades.
- **Singleton** asegura un acceso controlado a recursos compartidos.
- **Factory Method** desacopla la creación de objetos de su uso.
- **Template Method** estandariza la presentación de la interfaz.

Esta integración fortalece la calidad del software, facilitando su mantenimiento, escalabilidad y comprensión, aspectos fundamentales en el desarrollo de sistemas web modernos.

E. Beneficios de la Integración de Patrones en MVC

La integración de los patrones de diseño en la arquitectura MVC del sistema ConectaPet aporta los siguientes beneficios:

- Mayor modularidad del sistema.
- Reducción del acoplamiento entre componentes.
- Reutilización eficiente del código.
- Facilidad para incorporar nuevas funcionalidades.
- Mejora en la mantenibilidad y documentación del sistema.

C. Diseño de la Base de Datos

La base de datos fue diseñada bajo un modelo relacional, contemplando las entidades principales del sistema:

- Usuarios
- Mascotas
- Solicitudes de adopción
- Donaciones
- Noticias
- Seguimientos

Cada tabla fue normalizada para evitar redundancia de datos y asegurar la integridad referencial mediante claves primarias y foráneas.

E. Modelado UML del Sistema

Para el análisis y diseño del sistema se utilizaron diversos diagramas UML, entre ellos:

- **Diagramas de Casos de Uso:** Identifican las interacciones entre los actores y el sistema.
- **Diagramas de Secuencia:** Representan el flujo de mensajes entre vistas, controladores y modelos.
- **Diagramas de Estados:** Modelan el ciclo de vida de entidades como Mascota, Donación y Solicitud de Adopción.
- **Diagramas de Paquetes:** Organizan los módulos del sistema según su funcionalidad.

Estos diagramas permiten una comprensión clara del comportamiento dinámico y estructural del sistema.

4. RESULTADOS

Como resultado del desarrollo del sistema ConectaPet, se obtuvo una plataforma web funcional orientada a la adopción responsable de mascotas y a la gestión de donaciones. El sistema fue implementado bajo la arquitectura Modelo–Vista–Controlador(MVC), permitiendo una adecuada separación entre la lógica de negocio, la presentación y el control de las solicitudes.

La interfaz principal del sistema permite a los usuarios visualizar de manera clara las mascotas disponibles para adopción, facilitando la navegación y el acceso a la información relevante. En la **Fig. 9** se muestra la página principal del sistema ConectaPet, donde se presenta el listado de mascotas registradas junto con sus características básicas.

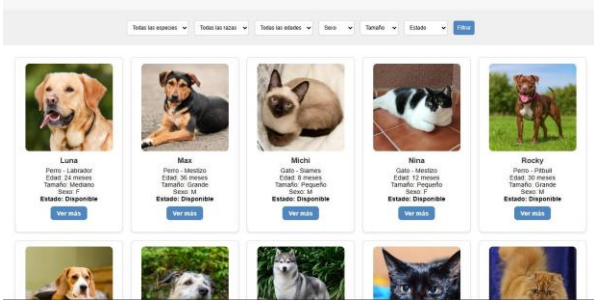

Fig. 9. *Interfaz principal del sistema ConectaPet mostrando el listado de mascotas disponibles para adopción.*

El proceso de adopción se gestiona mediante un formulario que permite al usuario registrar una solicitud asociada a una mascota específica. Este proceso es controlado por los controladores del sistema, los cuales validan la información ingresada y gestionan el flujo de datos entre las vistas y los modelos. La **Fig. 10** presenta el formulario de registro de solicitudes de adopción implementado en el sistema.

Fig. 10. *Formulario de registro de solicitudes de adopción del sistema ConectaPet.*

En cuanto a la gestión de donaciones, el sistema ofrece un módulo que permite a los usuarios realizar aportes destinados al bienestar de las mascotas, tales como donaciones de alimentos, medicinas o apoyo a refugios. Este módulo garantiza el registro ordenado de las donaciones y su posterior administración. La **Fig. 11** muestra la interfaz correspondiente al módulo de donaciones del sistema ConectaPet.

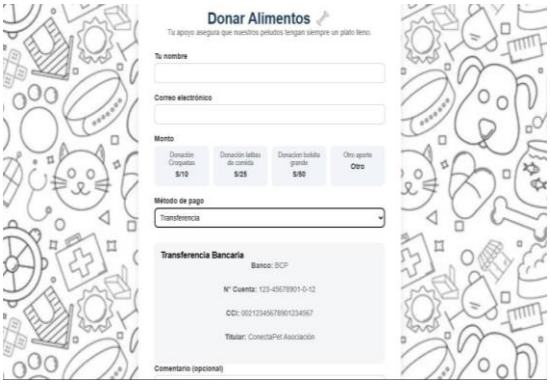

Fig. 11. *Interfaz del módulo de donaciones del sistema ConectaPet para el registro de aportes de los usuarios.*

Asimismo, se implementó un panel administrativo orientado a los administradores del sistema, desde el cual es posible gestionar mascotas, usuarios, donaciones y noticias. Este módulo permite realizar operaciones de creación, edición y eliminación de registros, asegurando el control de los procesos internos del sistema. En la **Fig. 12** se observa el panel administrativo desarrollado para la gestión de la información.

Fig. 12. *Panel administrativo del sistema ConectaPet para la gestión de mascotas, usuarios y donaciones.*

En conjunto, los resultados obtenidos evidencian que el sistema ConectaPet cumple con los objetivos propuestos, ofreciendo una solución web organizada, funcional y escalable, apoyada en el uso de la arquitectura MVC y patrones de diseño que fortalecen su estructura interna.

IV. ANÁLISIS DE RESULTADOS

A. Impacto de la arquitectura MVC en ConectaPet

La incorporación del patrón arquitectónico Modelo-Vista-Controlador (MVC) en ConectaPet permitió organizar el sistema de manera clara y estructurada. La separación entre la lógica de negocio, la capa de presentación y la interacción con el usuario generó un código más legible y sencillo de mantener. Este tipo de diseño resulta especialmente útil en aplicaciones web que requieren crecer y adaptarse a nuevas demandas, ya que disminuye la dependencia entre módulos y facilita la integración de funcionalidades adicionales [17]. En consecuencia, la arquitectura implementada no solo responde a los objetivos técnicos iniciales, sino que también se ajusta a prácticas consolidadas en el desarrollo de plataformas comunitarias.

B. Contribución de los patrones de diseño complementarios

La aplicación de patrones adicionales fortaleció la estabilidad del sistema:

- **Singleton:** aseguró una única instancia de conexión con la base de datos, evitando duplicaciones y optimizando el uso de recursos.
- **Factory Method:** permitió centralizar la creación de entidades como Mascota y Donación, reduciendo el acoplamiento entre capas y ofreciendo mayor flexibilidad para futuras ampliaciones.
- **Template Method:** garantizó uniformidad en las vistas, de modo que todas las páginas compartieran una misma estructura visual y se redujera la redundancia de código.

Estos elementos contribuyeron a la coherencia interna del sistema y facilitaron su mantenimiento, en concordancia con estudios que destacan la utilidad de los patrones en plataformas de gestión digital [18].

C. Ventajas y limitaciones observadas

Entre los beneficios más relevantes se identifican la modularidad, la reutilización de componentes y la claridad en la interfaz. Estas características no solo mejoran la experiencia del usuario, sino que también incrementan la confianza en la estabilidad del sistema. No obstante, la incorporación de múltiples patrones generó un aumento en la complejidad estructural y en el número de archivos, lo que requiere un nivel de experiencia adecuado para su correcta implementación. Este aspecto coincide con investigaciones que advierten sobre la necesidad de disciplina y buenas prácticas al aplicar arquitecturas basadas en MVC [19].

D. Interpretación general

El análisis realizado demuestra que ConectaPet no se limita a cumplir con los objetivos técnicos iniciales, sino que se consolida como una solución robusta y preparada para evolucionar. La combinación de MVC con patrones complementarios permitió construir una plataforma organizada, escalable y coherente, capaz de responder a las demandas de un sistema de adopción y donaciones en línea. En conjunto, los resultados evidencian que ConectaPet se alinea con experiencias previas en sistemas web comunitarios, consolidándose como una herramienta tecnológica sostenible y adaptable en el tiempo [20].

V. CONCLUSIÓN

En el presente artículo se abordó la aplicación de patrones de diseño en el desarrollo del sistema ConectaPet, evidenciando que el uso adecuado de estos patrones contribuye significativamente a la calidad del software. La implementación del patrón Singleton permitió una gestión eficiente de recursos compartidos, mientras que el patrón Factory Method favoreció la extensibilidad y desacoplamiento de los componentes del sistema.

Los resultados obtenidos demuestran que la aplicación de patrones de diseño no solo mejora la estructura interna del software, sino que también facilita su mantenimiento y evolución. En este sentido, se concluye que el uso de buenas prácticas de diseño resulta fundamental para el desarrollo de sistemas robustos y escalables en entornos reales.

VI. RECONOCIMIENTO

Los autores expresan su agradecimiento al docente José Arturo Díaz Pulido por la orientación académica y el acompañamiento brindado durante el desarrollo del presente trabajo. Asimismo, se agradece a la Universidad Nacional de Trujillo por

proporcionar el entorno académico que hizo posible la realización de este artículo.

VII. REFERENCIAS BIBLIOGRÁFICAS

[1]Petfinder Foundation. (2023). *Petfinder: Adopt a pet*. https://www.petfinder.com

[2]Adopt-a-Pet. (2023). *Adopt-a-Pet: Pet adoption and rescue*. https://www.adoptapet.com

[3]Rescate Animal Chile. (2022). *Rescate Animal Chile*. https://www.rescateanimalchile.org

[4] "Dominando el patrón Singleton en la arquitectura de software: eficiencia y acceso global," Curate Partners. https://curatepartners.com/tech-skills-tools-platforms/mastering-the-singleton-pattern-in-software-architecture-efficiency-and-global-access/

[5] "Dominando el patrón Singleton en la arquitectura de software: eficiencia y acceso global," Curate Partners. https://curatepartners.com/tech-skills-tools-platforms/mastering-the-singleton-pattern-in-software-architecture-efficiency-and-global-access/

[6] IRJMETS, "Exploring the Role of Design Patterns in Enhancing ASP.NET Core MVC Architecture," *International Research Journal of Modernization in Engineering Technology and Science*, Aug. 2024. https://www.irjmets.com/uploadedfiles/paper/issue_8_august_2024/61066/final/fin_irjmets1724169798.pdf

[7] Stack Overflow, "Factory pattern with 4 layer architecture," *Stack Overflow*, 2012. https://stackoverflow.com/questions/10630008/factory-pattern-with-4-layer-architecture

[8] Stack Overflow, "Where should we use Template Method pattern?," *Stack Overflow*, 2016. https://stackoverflow.com/questions/1553856/where-should-we-use-template-method-pattern

[9] Momentslog, "The Singleton Pattern in Database Connection Management," *Momentslog*, 2024. https://www.momentslog.com/development/design-pattern/the-singleton-pattern-in-database-connection-management

[10] Stack Overflow, "Is using the singleton pattern a good idea when accessing libraries using MVC?," *Stack Overflow*, 2015. https://stackoverflow.com/questions/29349244/is-using-the-singleton-pattern-a-good-idea-when-accessing-libraries-using-mvc

[11] GeeksforGeeks, "Factory Method Design Pattern," *GeeksforGeeks*, 2015. https://www.geeksforgeeks.org/system-design/factory-method-for-designing-pattern/

[12] GeeksforGeeks, "Singleton Design Pattern," *GeeksforGeeks*, 2016. [Online]. https://www.geeksforgeeks.org/system-design/singleton-design-pattern/

[13] Refactoring Guru, "Factory Method," *Refactoring Guru*, 2024. https://refactoring.guru/design-patterns/factory-method

[14] AI Publications, "The Role of Design Patterns in Code Reusability and Software Maintainability," *International Journal of Engineering, Business and Management*, Mar. 2025. https://aipublications.com/uploads/issue_files/7IJEBM-MAR20253-Enhancing.pdf

[15] Syracuse University, "Design Pattern Summary," *Department of Electrical Engineering and Computer Science*, Syracuse University. https://ecs.syr.edu/faculty/fawcett/handouts/CSE776/presentations/summary.pdf

[16] VEMU Institute of Technology, *"Lecture Notes on Design Patterns,"* 2020. https://vemu.org/uploads/lecture_notes/04_01_2020_1166301382.pdf

[17]J. P. Hidalgo Jácome and A. I. Usiña Follaran, Sistema Web de Gestión para adopción y recuperación de mascotas extraviadas para la Fundación Fauna Urbana Ibarra, PUCE-Ibarra, Ecuador, 2025. https://repositorio.puce.edu.ec/server/api/core/bitstreams/66dbe117-0533-49d6-beed-01f8ea2b407e/content

[18]A. Aguilar Domingo, Diseño e implementación de una aplicación web para gestión de adopción de mascotas gestionadas por albergues, protectoras y particulares, Univ. Miguel Hernández de Elche, España, 2023. https://dspace.umh.es/bitstream/11000/30229/1/TFG-Aguilar%20Domingo%2C%20Alberto.pdf

[19]D. Parra, "Patrones de diseño - Template Method", ThePowerUps Learning, 2022. https://thepowerups-learning.com/patrones-de-diseno-template-method

[20]B. Vargas, "Guía completa sobre el patrón Singleton: casos de uso y mejores prácticas", ByronVargas.com, 2024. https://www.byronvargas.com/web/cuando-usar-el-patron-singleton