**Universidad Centroamericana José Simeón Cañas**

**Facultad de Ingeniería y Arquitectura**

**Departamento de Electrónica e informática**


**Ingeniería de Software**

# Plan de pruebas – VetiCare

Catedrático:

Ing. Nelson Giovanni Castro Rodas

Ing. Luisa Daniela Arévalo Rodas

Nombre de integrantes:

Alison Aracely Argueta Flores 00076422 sección 01

Natalia Melissa Avelar Menjivar 00135922 sección 01

Diego Eduardo Castro Quintanilla 00117322 sección 01

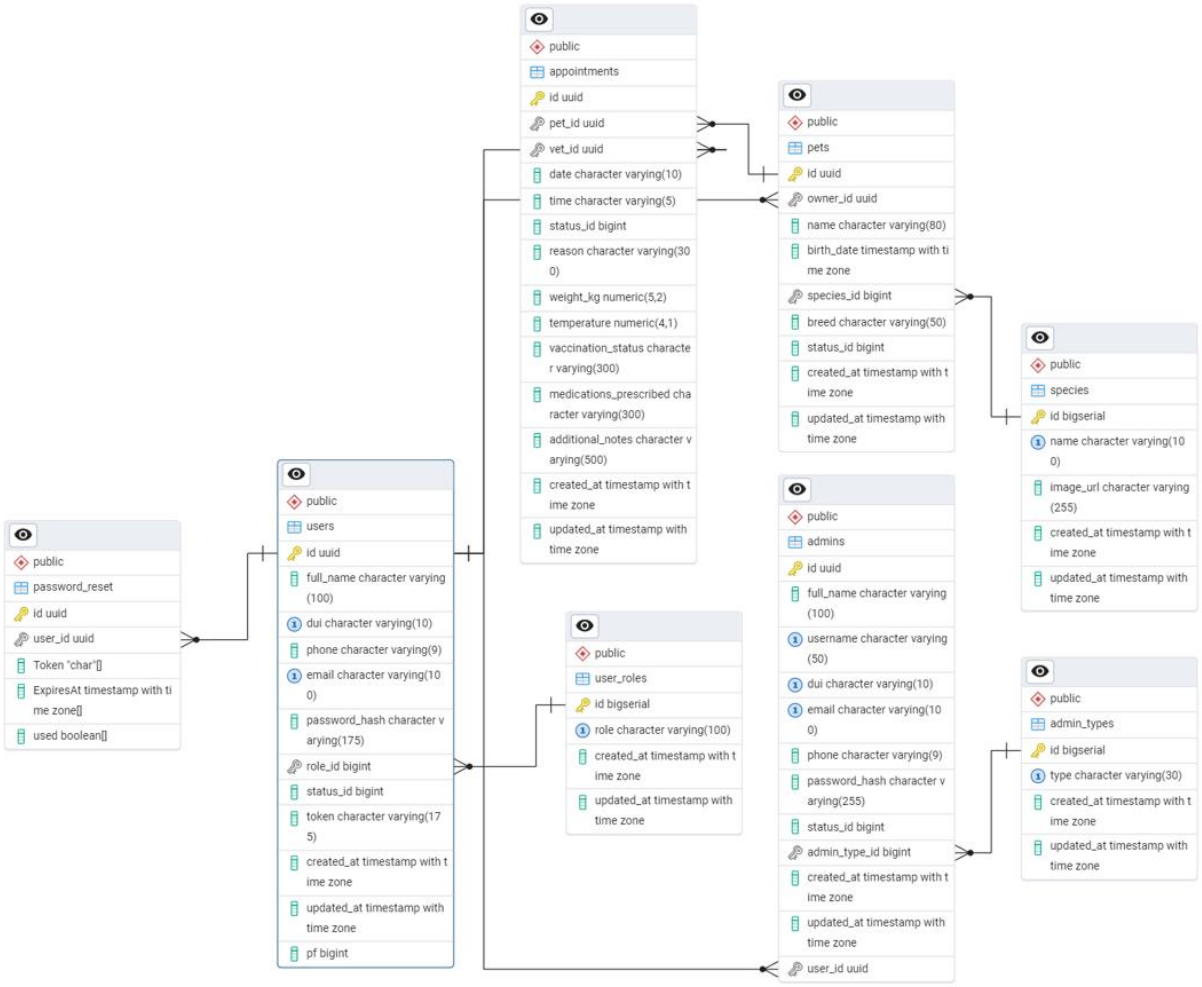
Josué Fernando Gómez Guardado 00103722 sección 01

Christian Joel López Ortega 00179320 sección 01

**Fecha: 10 de noviembre de 2025.**

## 2. Documentación de arquitectura

Diagrama de base de datos




## 4. Proceso de Desarrollo / Contribución

(Branching Strategy del Proyecto VetiCare)

Este proyecto utiliza una estrategia basada en git flow simplificado, adaptada al uso de ramas por historia de usuario.
El objetivo es mantener el código ordenado, minimizar conflictos y asegurar que cada cambio esté asociado a una funcionalidad clara.

### 4.1 Flujo general de trabajo

| Rama | Descripción |
|---|---|
| **main** | Contiene el código estable, probado y listo para producción. |
| **develop** | Rama base para las funcionalidades. Aquí se integran todas las historias de usuario completadas antes de ser movidas a producción. |

### 4.2. Ramas por Historia de Usuario

Cada desarrollador crea una rama exclusiva para la historia que está implementando.
El nombre sigue el formato:

feature/HU-<número> o feature/US-<número>

### 4.3. Pasos del flujo de desarrollo

**1. Crear rama desde develop**

Cada vez que un desarrollador comienza una historia:

1. git checkout develop
2. git pull
3. git checkout -b feature/HU-7-create-medical-record

**2. Desarrollo de la historia de usuario**

- Todo el código se implementa dentro de su rama correspondiente.
- Los commits deben ser claros y descriptivos, siendo permito utilizar gitmoji:

feat(HU-7): “agregar formulario para creación de expediente”

fix(HU-4): “hacer visible botón de cierre de modal”

**3. Pruebas individuales**

Antes de subir código, cada desarrollador debe:

✓ Probar funcionalidad
✓ Verificar que no se rompen flujos existentes

✓ Revisar consola del navegador
✓ Ejecutar pruebas manuales (según el plan de pruebas del documento)

**4. Subir la rama al repositorio**

- git add .
- git commit -m "feat(HU-(numero): guardar expediente médico correctamente"
- git push --set-upstream origin feature/HU-(numero)-create-medical-record

**5. Solicitud de Pull Request (PR)**

Cada historia **debe** pasar por una revisión en GitHub antes de fusionarse a develop.

El PR debe incluir:

- Historia de usuario asociada
- Código bien comentado
- Checklist de criterios de aceptación

**6. Code Review (Revisión por otro compañero)**

Otro integrante revisa:

- Código limpio
- Manejo correcto de errores
- Funcionalidad intacta
- No rompe otras partes del sistema
- Cumple criterios de aceptación

**7. Merge hacia develop**

Se realiza **solo cuando el PR está aprobado**.

**8. Pruebas integradas**

Al finalizar un sprint o conjunto de historias:

✓ Se prueba la rama develop completa
✓ Se validan flujos completos por rol (dueño, veterinario y administrador)
✓ Se revisa que nada rompió el comportamiento ya existente

**9. Deploy a producción**

Cuando develop está estable:

- Se fusiona a main
- Se etiqueta la versión (ejemplo: `v1.0.3`)
- Se despliega

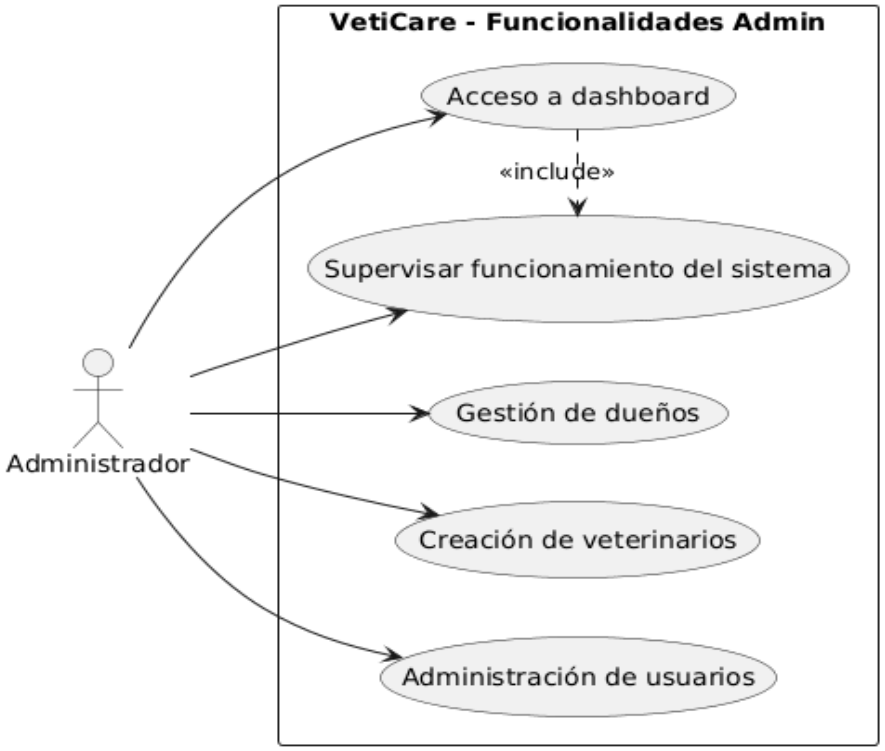
### 4.4. Buenas prácticas del equipo

- Cada desarrollador solo trabaja en su rama.
- Nunca se hace commit directo a develop o main.
- Las ramas de feature se eliminan después de fusionarse.
- Todo commit debe estar relacionado a una historia de usuario.
- Los conflictos se resuelven en la rama feature antes del merge.

DOCUMENTACION DE LOGICA DEL NEGOCIO:

### 1. Casos de usos – Actores y sus interacciones con el sistema.


VetiCare - Funcionalidades Dueño
Agendar nueva cita
«include»
Consultar citas (reservadas)
«include»
Consultar historial de citas
Registrarse en el sistema
Registrar mascota
Gestionar mascotas (editar y eliminar)
Editar perfil
Ver historial médico de cada mascota
Dueño de mascota


## 2. Reglas de Negocio de VetiCare

Aquí te dejo un conjunto sólido, claro y formal de reglas de negocio según tu aplicación.

## Reglas de Negocio

### RN-01. Registro de Usuarios

Todo usuario (dueño o veterinario) debe poseer un correo único.
No se permite el registro de dos usuarios con el mismo email o DUI.

### RN-02. Roles y Permisos

Los roles determinan lo que un usuario puede hacer:

- Rol Dueño: puede ver sus mascotas, crear citas, ver historial.
- Rol Veterinario: puede ver citas, registrar nuevas mascotas (nuevos expedientes) y editar expedientes.
- Rol Administrador: puede gestionar usuarios, mascotas, veterinarios, especies.

### RN-03. Mascotas por Usuario

Cada mascota debe estar asociada únicamente a un dueño.
No puede existir una mascota sin dueño registrado.

### RN-04. Registro de Citas

Un dueño solo puede crear citas para mascotas que le pertenecen.
Una cita debe contar con:

- fecha
- hora
- Mascota
- Nombre

### RN-05. Disponibilidad del Veterinario

No pueden asignarse dos citas para el mismo veterinario a la misma hora.

### RN-06. Estado de la Cita

Las citas solo pueden estar en los siguientes estados:

- Programada
- En proceso

- Finalizada
- Cancelada

Los cambios deben registrarse en la BD.

### RN-07. Expediente Clínico

Cada cita finalizada debe generar un registro clínico con:

- peso
- temperatura
- medicamentos recetados
- notas adicionales
- estado de vacunación

### RN-08. Especies

Las mascotas solamente pueden ser de especies registradas:

- Perro
- Gato
- Ave

No se permiten especies fuera de catálogo.

### RN-09. Seguridad

Toda acción dentro del sistema requiere un token JWT válido.
Si el token expira, el sistema debe cerrar sesión automáticamente.

## 3. Glosario de Términos

Diseñado según el dominio de VetiCare.

### Glosario VetiCare

### Usuario

Persona registrada en el sistema. Puede ser dueño, veterinario o administrador.

### Dueño

Usuario responsable de una o más mascotas. Puede agendar citas y ver el historial médico.

### Veterinario

Usuario con permisos para atender citas y actualizar expedientes.

### Administrador

Usuario con control total del sistema:
gestiona especies, veterinarios, dueños, y citas.

### Mascota

Animal registrado en el sistema, asociado obligatoriamente a un dueño.

### Especie

Categoría de la mascota (perro, gato, ave).

### Cita

Registro de una consulta veterinaria programada. Incluye fecha, hora, mascota y veterinario.

### Estado de la Cita

Define el progreso de la cita (programada, finalizada, cancelada).

### Expediente Clínico

Historial médico de una mascota generado a partir de citas atendidas.

### Token JWT

Código de seguridad generado al iniciar sesión, necesario para realizar peticiones a la API.

### GORM

ORM usado por el backend Go para interactuar con PostgreSQL.

## API REST

Interfaz que permite la comunicación entre el frontend y el backend de manera estructurada.