****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto *“Sistema Web de Gestión Veterinaria para Mascotas”***

Curso: *Programación Web II*

Docente: *Ing. Patrick Cuadros Quiroga*

Integrantes:

***Ccalli Chata, Joel Robert (2017057528)***

***Apaza Mamani, Edward Hernan (2018060915)***

**Tacna – Perú**

***2025***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | JCC,EPM | JCC,EPM | JCC,EPM | 11/02/2025 | Versión Original |

Sistema *Web de Gestión Veterinaria para Mascotas*

Documento de Arquitectura de Software

Versión *1.0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | JCC,EPM | JCC,EPM | JCC,EPM | 11/02/2025 | Versión Original |

Contenido

[1. INTRODUCCIÓN 5](#_Toc69808834)

[1.1. Propósito (Diagrama 4+1) 5](#_Toc69808835)

[1.2. Alcance 5](#_Toc69808836)

[1.3. Definición, siglas y abreviaturas 5](#_Toc69808837)

[1.4. Organización del documento 5](#_Toc69808838)

[2. OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS 5](#_Toc69808839)

[2.1.1. Requerimientos Funcionales 5](#_Toc69808840)

[2.1.2. Requerimientos No Funcionales – Atributos de Calidad 5](#_Toc69808841)

[3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA 6](#_Toc69808842)

[3.1. Vista de Caso de uso 6](#_Toc69808843)

[3.1.1. Diagramas de Casos de uso 6](#_Toc69808844)

[3.2. Vista Lógica 6](#_Toc69808845)

[3.2.1. Diagrama de Subsistemas (paquetes) 7](#_Toc69808846)

[3.2.2. Diagrama de Secuencia (vista de diseño) 7](#_Toc69808847)

[3.2.3. Diagrama de Colaboración (vista de diseño) 7](#_Toc69808848)

[3.2.4. Diagrama de Objetos 7](#_Toc69808849)

[3.2.5. Diagrama de Clases 7](#_Toc69808850)

[3.2.6. Diagrama de Base de datos (relacional o no relacional) 7](#_Toc69808851)

[3.3. Vista de Implementación (vista de desarrollo) 7](#_Toc69808852)

[3.3.1. Diagrama de arquitectura software (paquetes) 7](#_Toc69808853)

[3.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes) 7](#_Toc69808854)

[3.4. Vista de procesos 7](#_Toc69808855)

[3.4.1. Diagrama de Procesos del sistema (diagrama de actividad) 8](#_Toc69808856)

[3.5. Vista de Despliegue (vista física) 8](#_Toc69808857)

[3.5.1. Diagrama de despliegue 8](#_Toc69808858)

[4. ATRIBUTOS DE CALIDAD DEL SOFTWARE 8](#_Toc69808859)

[Escenario de Funcionalidad 8](#_Toc69808860)

[Escenario de Usabilidad 8](#_Toc69808861)

[Escenario de confiabilidad 9](#_Toc69808862)

[Escenario de rendimiento 9](#_Toc69808863)

[Escenario de mantenibilidad 9](#_Toc69808864)

[Otros Escenarios 9](#_Toc69808865)

**1. INTRODUCCIÓN**

**1.1 Propósito (Diagrama 4+1)**

El propósito de este documento es describir la arquitectura del Sistema Web de Gestión Veterinaria para Mascotas utilizando el modelo 4+1 vistas, proporcionando una representación clara y estructurada del sistema.

**1.2 Alcance**

El sistema cubrirá funcionalidades como:

* Registro de usuarios y mascotas.
* Programación y gestión de citas veterinarias.
* Notificaciones automáticas de recordatorio.
* Almacenamiento y consulta de historiales médicos de mascotas. Estas características permitirán a veterinarias y dueños de mascotas optimizar sus procesos y mejorar la comunicación.

**1.3 Definición, siglas y abreviaturas**

* **SAD**: Software Architecture Document
* **MVC**: Modelo Vista Controlador
* **DB**: Base de Datos
* **RF**: Requerimiento Funcional
* **RNF**: Requerimiento No Funcional

**1.4 Organización del documento**

Este documento se divide en secciones que describen los objetivos arquitectónicos, representaciones visuales del sistema y atributos de calidad del software.

**2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS**

**2.1.1 Requerimientos Funcionales**

|  |  |  |
| --- | --- | --- |
| **ID** | **Requerimiento** | **Prioridad** |
| RF1 | Registro de usuarios | Alta |
| RF2 | Gestión de citas | Alta |
| RF3 | Notificaciones automáticas | Media |
| RF4 | Consulta de historiales | Alta |
| RF5 | Gestionar productos | Media |
| RF6 | Generación de reportes | Baja |

**2.1.2 Requerimientos No Funcionales – Atributos de Calidad**

|  |  |  |
| --- | --- | --- |
| ID | Requerimiento | Prioridad |
| RNF1 | Disponibilidad del 99.9% | Alta |
| RNF2 | Tiempo de respuesta < 3 segundos | Alta |
| RNF3 | Compatibilidad con navegadores | Media |
| RNF4 | Seguridad mediante cifrado bcrypt | Alta |
| RNF5 | Escalabilidad para 10,000 usuarios | Media |

**3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA**

**3.1. Vista de Caso de Uso**

**3.1.1. Diagramas de Casos de Uso**

@startuml

left to right direction

actor Administrador as Admin <<Administrador>>

rectangle "Sistema Web de Gestión Veterinaria" {

Admin --> (Iniciar Sesión)

Admin --> (Registrar Usuario)

Admin --> (Historial Médico de Mascotas)

Admin --> (Notificaciones de Vacunas)

Admin --> (Programar Citas)

Admin --> (Agregar Mascota)

Admin --> (Actualizar Perfil)

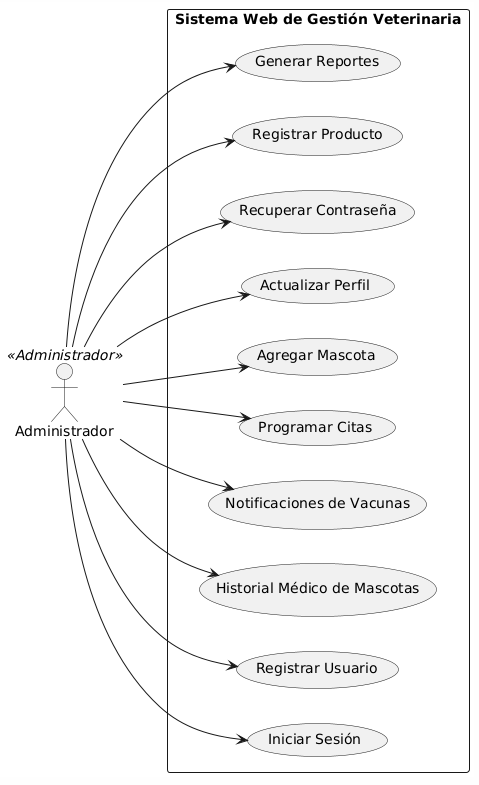
Admin --> (Recuperar Contraseña)

Admin --> (Registrar Producto)

Admin --> (Generar Reportes)

}

@enduml



**3.2. Vista Lógica**

**3.2.1. Diagrama de Subsistemas (Paquetes)**

Código en mermaid:

graph TD

subgraph "Sistema de Gestión Veterinaria"

A[Interfaz Usuario] -->|Solicita información| B[Gestión de Pacientes]

A -->|Accede a citas| C[Gestión de Citas]

A -->|Consulta historial| D[Gestión de Historias Clínicas]

A -->|Realiza pagos| E[Gestión de Pagos]

B -->|Guarda datos| F[Base de Datos]

C -->|Guarda y recupera citas| F

D -->|Registra y consulta historial| F

E -->|Procesa pagos| G[Pasarela de Pago]

end

subgraph "Módulos Adicionales"

H[Gestión de Usuarios] -->|Autenticación y permisos| A

I[Notificaciones] -->|Envía recordatorios| C

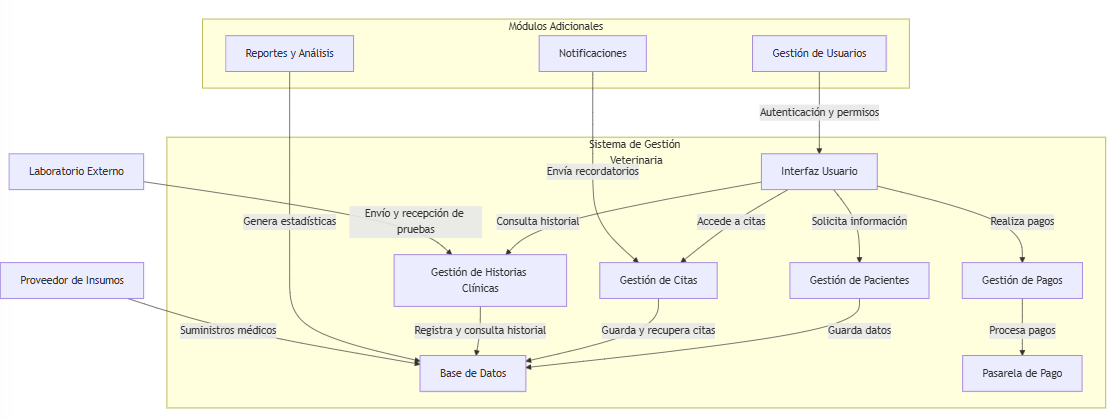
J[Reportes y Análisis] -->|Genera estadísticas| F

end

%% Conexiones externas

K[Proveedor de Insumos] -->|Suministros médicos| F

L[Laboratorio Externo] -->|Envío y recepción de pruebas| D



**3.2.2. Diagrama de Secuencia (Vista de Diseño)**

Mermaid:

sequenceDiagram

autonumber

participant Cliente 🧑‍💻

participant Sistema 🖥️

participant Veterinario 🩺

participant Diagnóstico 📄

Cliente->>+Sistema: 🔐 Iniciar sesión

Sistema-->>-Cliente: ✅ Autenticación exitosa

Cliente->>+Sistema: 📋 Registrar mascota

Sistema-->>-Cliente: 🐾 Mascota registrada con éxito

Cliente->>+Sistema: 📅 Solicitar consulta

Sistema-->>Cliente: 📆 Mostrar disponibilidad

Cliente->>Sistema: 🏥 Seleccionar fecha y veterinario

Sistema->>+Veterinario: 📩 Notificar nueva consulta

Veterinario-->>-Sistema: ✅ Confirmar disponibilidad

Sistema-->>Cliente: 📌 Confirmación de consulta

Cliente->>+Sistema: 📜 Consultar historial de consultas

Sistema-->>-Cliente: 📊 Mostrar consultas pasadas

Cliente->>+Sistema: 🚪 Asistir a la consulta

Sistema->>+Veterinario: 🔍 Registrar asistencia

Veterinario->>+Sistema: 📝 Registrar diagnóstico

Sistema->>+Diagnóstico: 💾 Guardar información médica

Diagnóstico-->>-Sistema: 📑 Diagnóstico almacenado

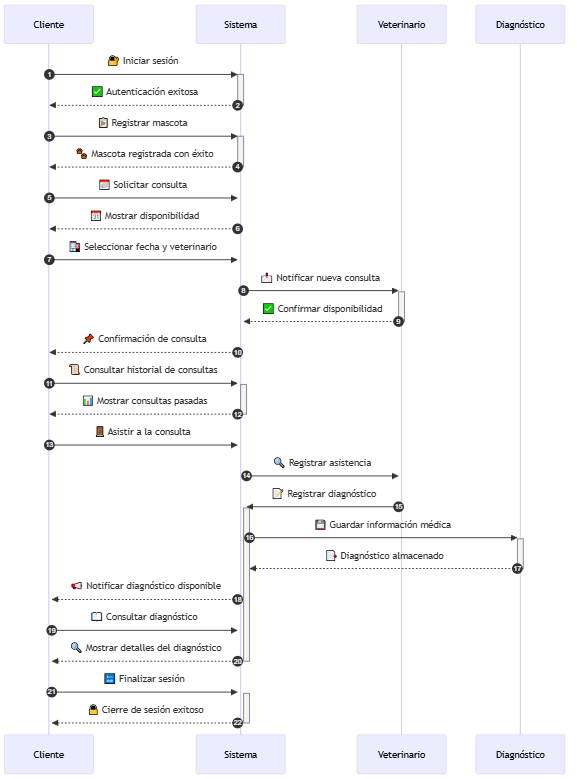
Sistema-->>+Cliente: 📢 Notificar diagnóstico disponible

Cliente->>Sistema: 📖 Consultar diagnóstico

Sistema-->>-Cliente: 🔍 Mostrar detalles del diagnóstico

Cliente->>+Sistema: 🔚 Finalizar sesión

Sistema-->>-Cliente: 🔒 Cierre de sesión exitoso



**3.2.3. Diagrama de Colaboración (Vista de Diseño)**

Mermaid:

graph TD

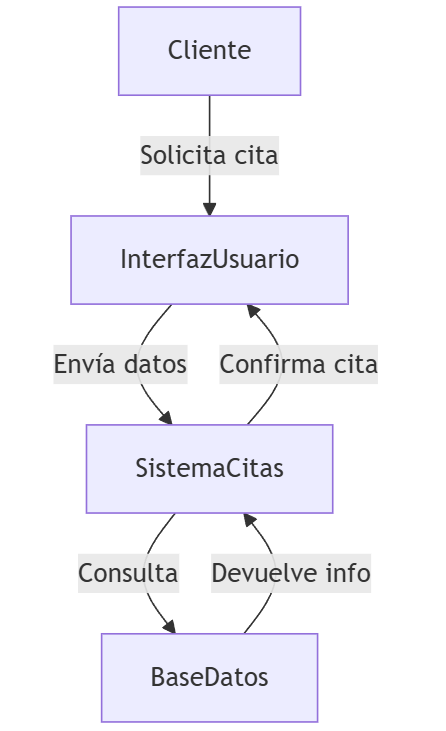
Cliente -->|Solicita cita| InterfazUsuario

InterfazUsuario -->|Envía datos| SistemaCitas

SistemaCitas -->|Consulta| BaseDatos

BaseDatos -->|Devuelve info| SistemaCitas

SistemaCitas -->|Confirma cita| InterfazUsuario



**3.2.4. Diagrama de Objetos**

PlantUML:

@startuml

left to right direction

object Cliente1 {

IdCliente = 1

Nombre = "Juan Pérez"

Dni = "12345678"

Email = "juan@email.com"

Direccion = "Av. Principal 123"

Telefono = "987654321"

}

object Mascota1 {

IdMascota = 101

Nombre = "Firulais"

Especie = "Perro"

Raza = "Labrador"

Edad = 5

Color = "Dorado"

IdCliente = 1

}

object Veterinario1 {

IdVeterinario = 201

Nombre = "Dra. María López"

Especialidad = "Medicina Interna"

Email = "maria@email.com"

Telefono = "987654322"

}

object Consulta1 {

IdConsulta = 301

IdMascota = 101

IdCliente = 1

IdVeterinario = 201

Fecha = "2025-02-11"

Hora = "10:00 AM"

Descripcion = "Revisión general"

}

object Diagnostico1 {

IdDiagnostico = 401

IdConsulta = 301

Peso = 25.5

Observaciones = "Buen estado de salud"

ExamenesRealizados = "Ninguno"

DiagnosticoGeneral = "Mascota saludable"

FechaCreacion = "2025-02-11"

}

object Usuario1 {

IdUsuario = 501

NombreUsuario = "admin"

Contrasena = "\*\*\*\*\*\*\*\*"

Rol = "Administrador"

}

Cliente1 -- Mascota1 : posee

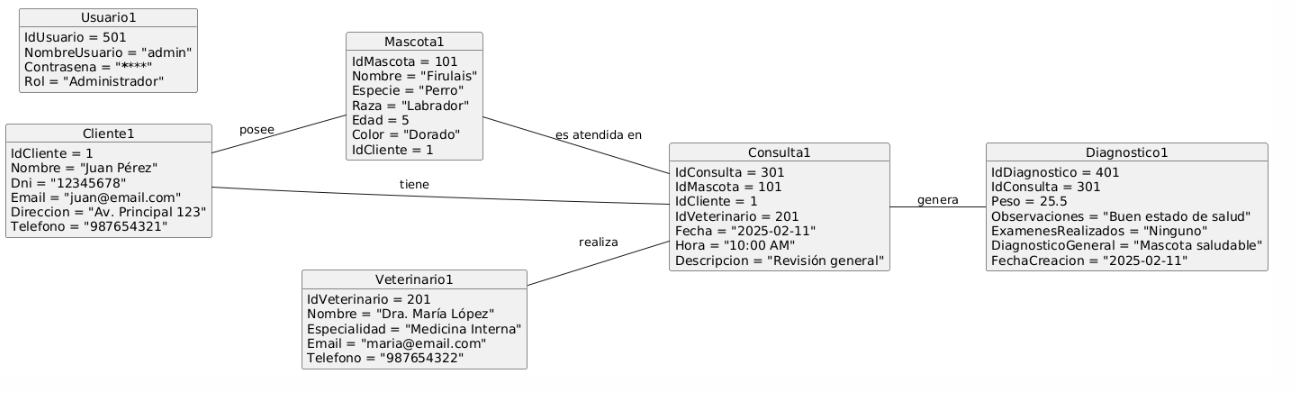
Cliente1 -- Consulta1 : tiene

Mascota1 -- Consulta1 : es atendida en

Veterinario1 -- Consulta1 : realiza

Consulta1 -- Diagnostico1 : genera

@enduml



**3.2.5. Diagrama de Clases**

PlanUML:

@startuml

class Cliente {

+IdCliente: int

+Nombre: string

+Dni: string

+Email: string

+Direccion: string

+Telefono: string

}

class Mascota {

+IdMascota: int

+Nombre: string

+Especie: string

+Raza: string

+Edad: int

+Color: string

+IdCliente: int

}

class Veterinario {

+IdVeterinario: int

+Nombre: string

+Especialidad: string

+Email: string

+Telefono: string

}

class Consulta {

+IdConsulta: int

+IdMascota: int

+IdCliente: int

+IdVeterinario: int

+Fecha: datetime

+Hora: string

+Descripcion: string

}

class Diagnostico {

+IdDiagnostico: int

+IdConsulta: int

+Peso: decimal

+Observaciones: string

+ExamenesRealizados: string

+DiagnosticoGeneral: string

+FechaCreacion: datetime

}

class Usuario {

+IdUsuario: int

+NombreUsuario: string

+Contrasena: string

+Rol: string

}

Cliente "1" -- "\*" Mascota : posee

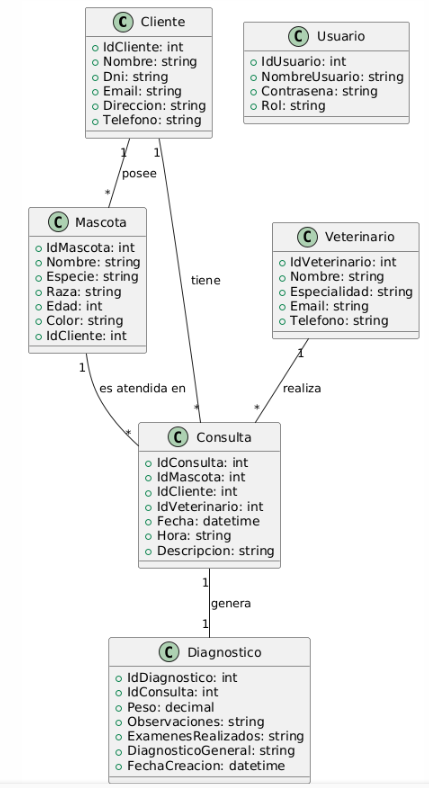
Cliente "1" -- "\*" Consulta : tiene

Mascota "1" -- "\*" Consulta : es atendida en

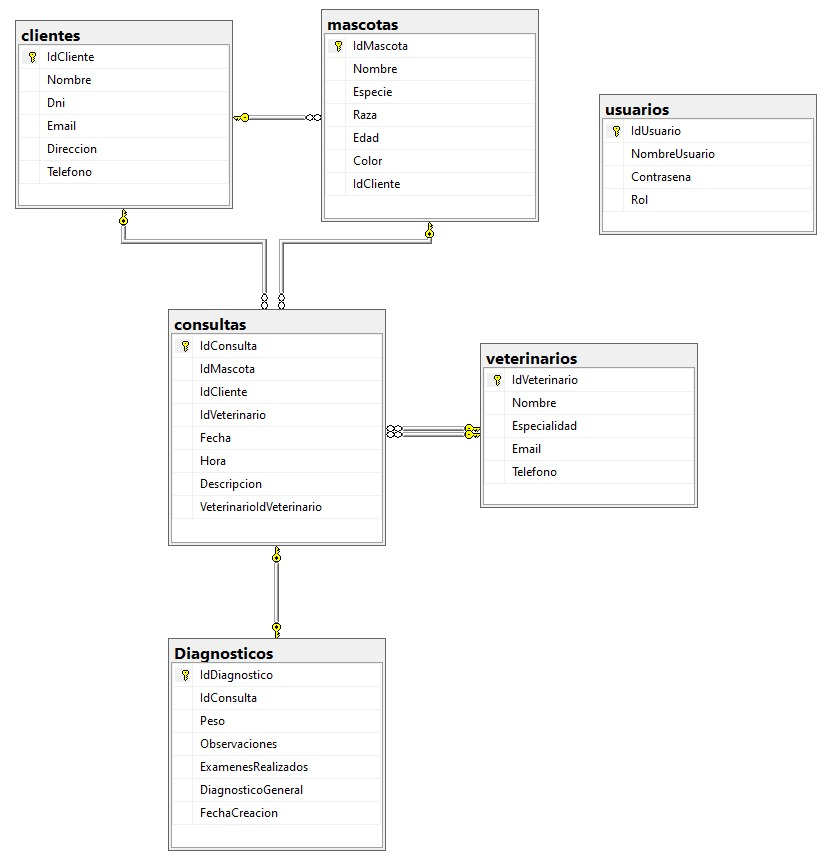
Veterinario "1" -- "\*" Consulta : realiza

Consulta "1" -- "1" Diagnostico : genera

@enduml



**3.2.6. Diagrama de Base de Datos**



**3.3. Vista de Implementación**

**3.3.1. Diagrama de Arquitectura Software (Paquetes)**

📌 **PlantUML**

plantuml

CopiarEditar

@startuml

package "Sistema" {

[UI]

[Lógica de Negocio]

[Persistencia]

}

UI --> Lógica de Negocio

Lógica de Negocio --> Persistencia

@enduml

📌 **Mermaid**

mermaid

CopiarEditar

graph TD

A[UI] --> B[Lógica de Negocio]

B --> C[Persistencia]

**3.3.2. Diagrama de Arquitectura del Sistema (Componentes)**

📌 **PlantUML**

plantuml

CopiarEditar

@startuml

component "Cliente Web" as CW

component "Servidor Aplicación" as SA

component "Base de Datos" as BD

CW --> SA

SA --> BD

@enduml

📌 **Mermaid**

mermaid

CopiarEditar

graph TD

A[Cliente Web] --> B[Servidor Aplicación]

B --> C[Base de Datos]

**3.4. Vista de Procesos**

**3.4.1. Diagrama de Actividad**

📌 **PlantUML**

plantuml

CopiarEditar

@startuml

start

:Usuario ingresa presupuesto;

:Sistema analiza opciones;

if (Recomendaciones disponibles?) then (Sí)

:Muestra opciones;

else (No)

:Muestra error;

endif

stop

@enduml

📌 **Mermaid**

mermaid

CopiarEditar

graph TD

A[Inicio] --> B[Ingresar presupuesto]

B --> C[Analizar opciones]

C -->|Sí| D[Mostrar opciones]

C -->|No| E[Mostrar error]

D --> F[Fin]

**4. ATRIBUTOS DE CALIDAD DEL SOFTWARE**

* **Escenario de Funcionalidad**: El sistema debe permitir la gestión de usuarios, citas y consultas veterinarias.
* **Escenario de Usabilidad**: Interfaz intuitiva para veterinarios y dueños de mascotas.
* **Escenario de Confiabilidad**: Disponibilidad del 99.9% y almacenamiento seguro.
* **Escenario de Rendimiento**: Tiempo de respuesta inferior a 3 segundos.
* **Escenario de Mantenibilidad**: Arquitectura modular para futuras mejoras.

Este documento proporciona una estructura bien organizada y detallada para el desarrollo del sistema web de gestión veterinaria.