



**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

<b>1. INTRODUCCIÓN .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>1.1. Propósito (Diagrama 4+1).....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>1.2. Alcance.....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>1.3. Definición, siglas y abreviaturas .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>1.4. Organización del documento .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>2. OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
2.1.1. Requerimientos Funcionales .....	<b><i>¡Error! Marcador no definido.</i></b>
2.1.2. Requerimientos No Funcionales – Atributos de Calidad.....	<b><i>¡Error! Marcador no definido.</i></b>
<b>3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>3.1. Vista de Caso de uso .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
3.1.1. Diagramas de Casos de uso .....	<b><i>¡Error! Marcador no definido.</i></b>
<b>3.2. Vista Lógica.....</b>	<b><i>¡Error! Marcador no definido.</i></b>
3.2.1. Diagrama de Subsistemas (paquetes) .....	<b><i>¡Error! Marcador no definido.</i></b>
3.2.2. Diagrama de Secuencia (vista de diseño).....	<b><i>¡Error! Marcador no definido.</i></b>
3.2.3. Diagrama de Colaboración (vista de diseño).....	<b><i>¡Error! Marcador no definido.</i></b>
3.2.4. Diagrama de Objetos.....	<b><i>¡Error! Marcador no definido.</i></b>
3.2.5. Diagrama de Clases .....	<b><i>¡Error! Marcador no definido.</i></b>
3.2.6. Diagrama de Base de datos (relacional o no relacional) .....	<b><i>¡Error! Marcador no definido.</i></b>
<b>3.3. Vista de Implementación (vista de desarrollo) .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
3.3.1. Diagrama de arquitectura software (paquetes).....	<b><i>¡Error! Marcador no definido.</i></b>
3.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes) .	<b><i>¡Error! Marcador no definido.</i></b>
<b>3.4. Vista de procesos .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
3.4.1. Diagrama de Procesos del sistema (diagrama de actividad)	<b><i>¡Error! Marcador no definido.</i></b>
<b>3.5. Vista de Despliegue (vista física).....</b>	<b><i>¡Error! Marcador no definido.</i></b>
3.5.1. Diagrama de despliegue .....	<b><i>¡Error! Marcador no definido.</i></b>
<b>4. ATRIBUTOS DE CALIDAD DEL SOFTWARE.....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>Escenario de Funcionalidad .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>Escenario de Usabilidad .....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>Escenario de confiabilidad.....</b>	<b><i>¡Error! Marcador no definido.</i></b>
<b>Escenario de rendimiento .....</b>	<b><i>¡Error! Marcador no definido.</i></b>



Escenario de mantenibilidad ..... ¡Error! Marcador no definido.

Otros Escenarios ..... ¡Error! Marcador no definido.

# 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

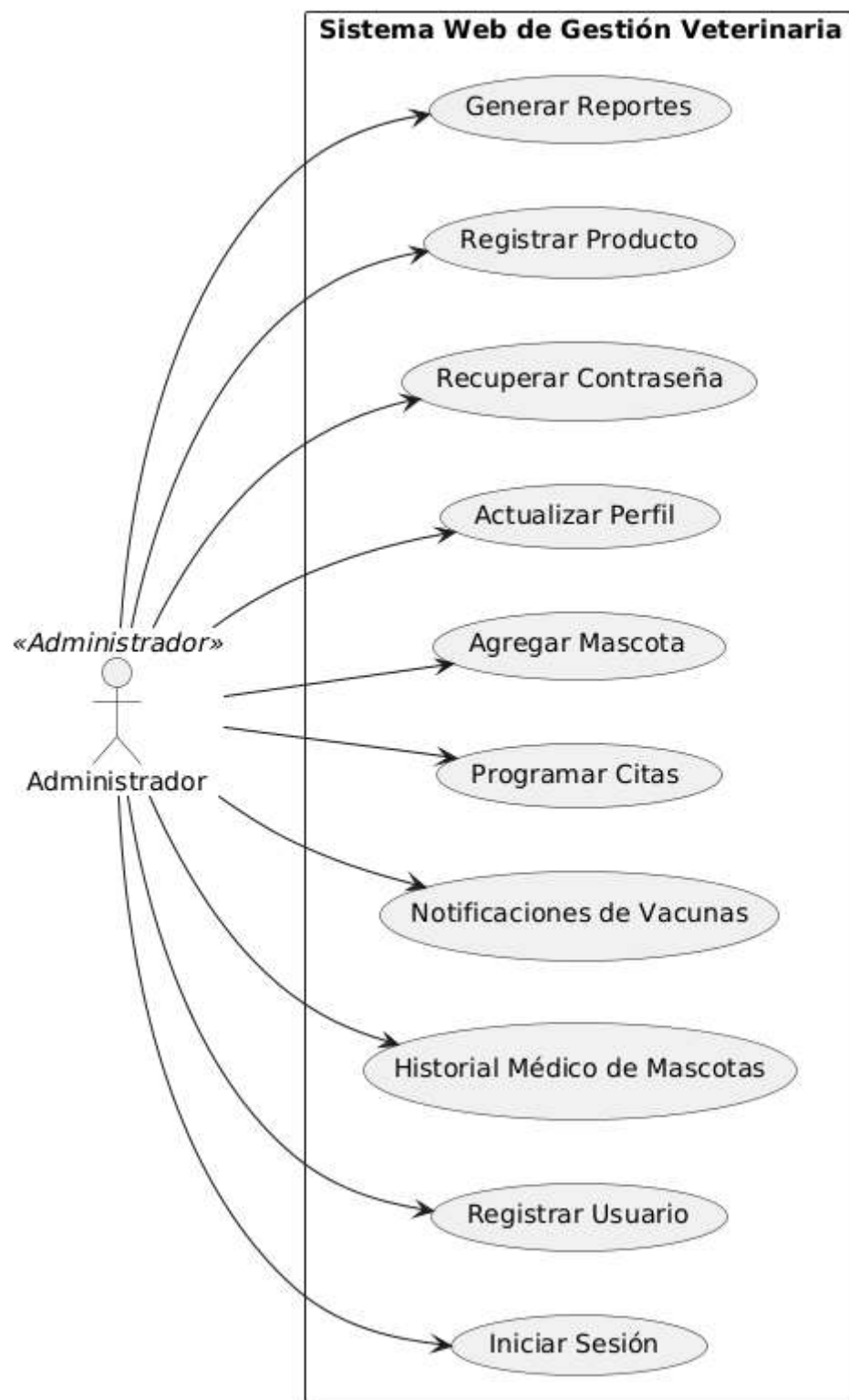
## 3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA

### 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)

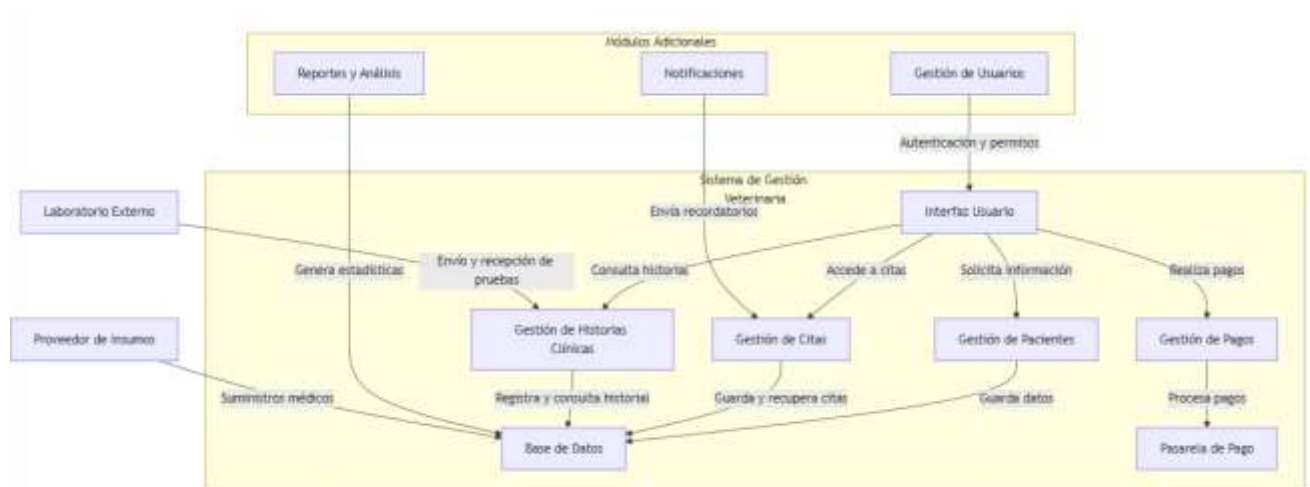
Código en mermaid:

```
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
    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

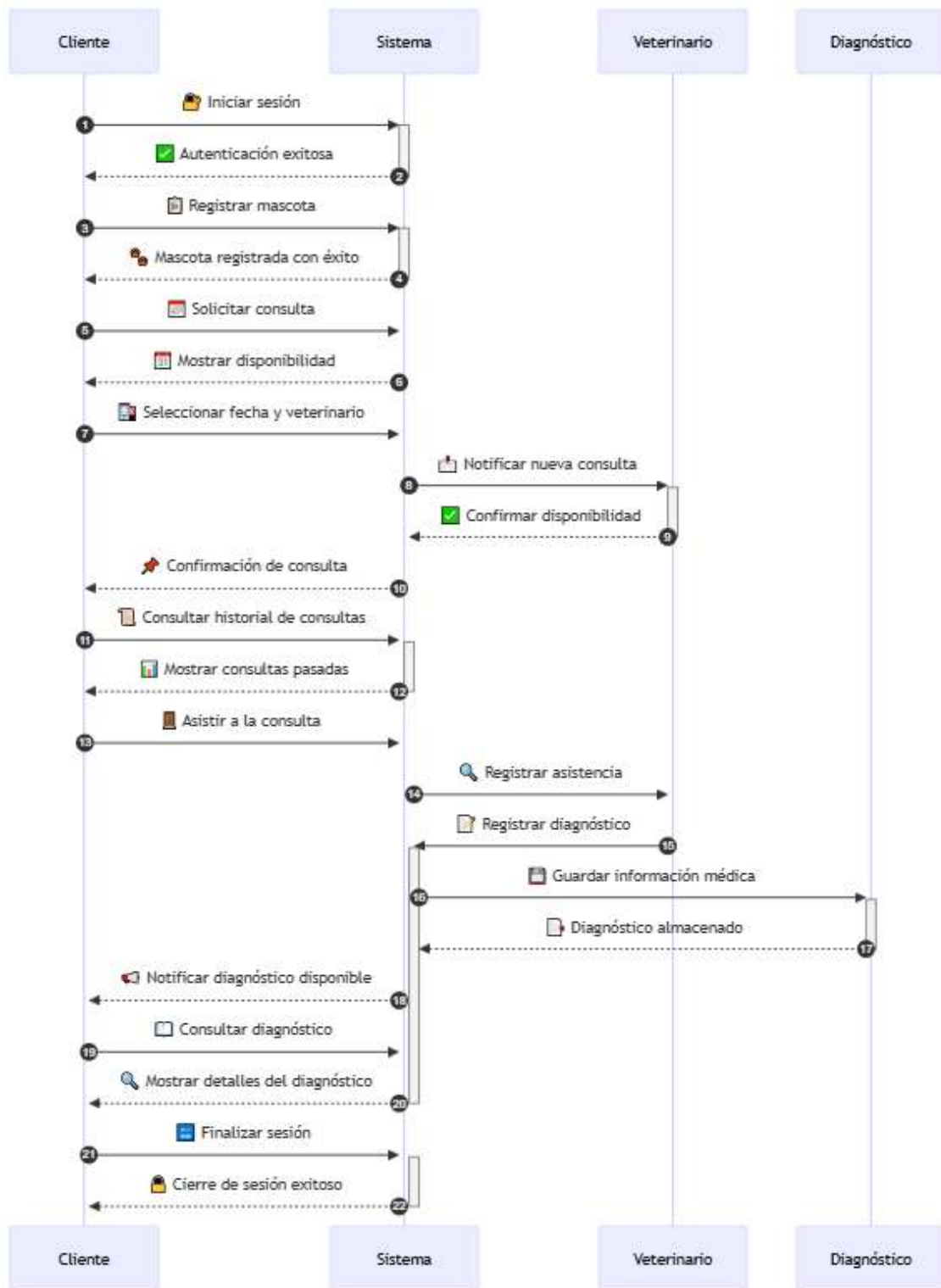
Cliente->>Sistema:  Consultar diagnóstico

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:



## classDiagram

%% Definición de Clases (Objetos)

```
class Cliente {
    +String nombre
    +String direccion
    +String telefono
    +List<Mascota> mascotas
    +registrarMascota()
    +solicitarConsulta()
}

class Mascota {
    +String nombre
    +String especie
    +int edad
    +HistorialMedico historial
    +List<Consulta> consultas
}

class Veterinario {
    +String nombre
    +String especialidad
    +List<Consulta> consultas
    +registrarDiagnostico()
}

class Consulta {
    +Date fecha
    +String motivo
    +Diagnostico diagnostico
    +registrarConsulta()
}

class Diagnostico {
    +String descripcion
    +String tratamiento
}

class Usuario {
    +String usuario
    +String contraseña
    +String rol
    +iniciarSesion()
    +cerrarSesion()
}
```



```
class Administrador {
    +String nombre
    +gestionarUsuarios()
    +gestionarProductos()
    +generarReportes()
}

class Producto {
    +String nombre
    +float precio
    +int stock
}

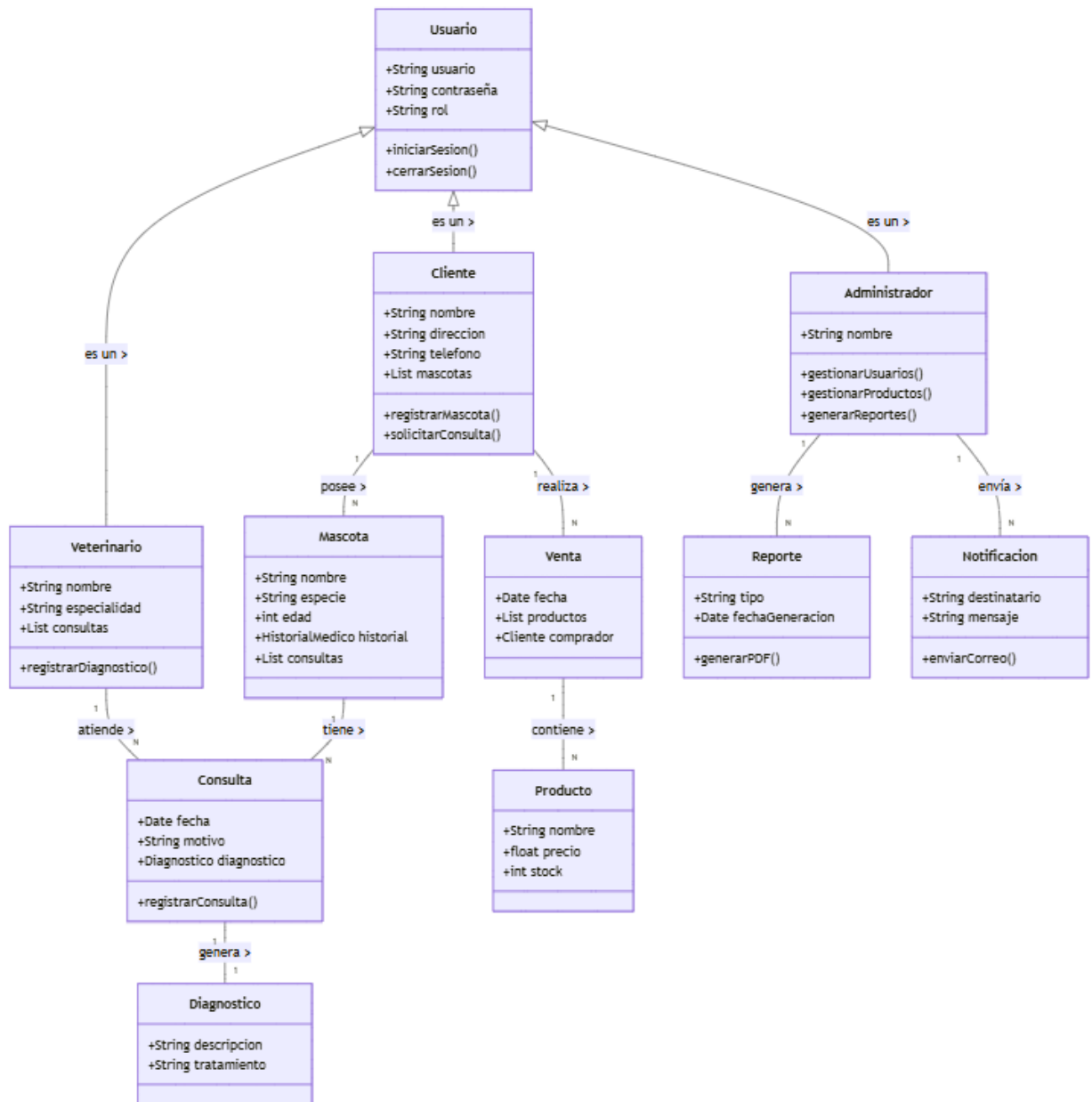
class Venta {
    +Date fecha
    +List<Producto> productos
    +Cliente comprador
}

class Reporte {
    +String tipo
    +Date fechaGeneracion
    +generarPDF()
}

class Notificacion {
    +String destinatario
    +String mensaje
    +enviarCorreo()
}

%% Relaciones entre objetos

Cliente "1" -- "N" Mascota : posee >
Mascota "1" -- "N" Consulta : tiene >
Consulta "1" -- "1" Diagnostico : genera >
Veterinario "1" -- "N" Consulta : atiende >
Usuario <|-- Cliente : es un >
Usuario <|-- Administrador : es un >
Usuario <|-- Veterinario : es un >
Administrador "1" -- "N" Reporte : genera >
Administrador "1" -- "N" Notificacion : envía >
Venta "1" -- "N" Producto : contiene >
Cliente "1" -- "N" Venta : realiza >
```



### 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"
  ExamenRealizados = "Ninguno"
  DiagnosticoGeneral = "Mascota saludable"
  FechaCreacion = "2025-02-11"
```

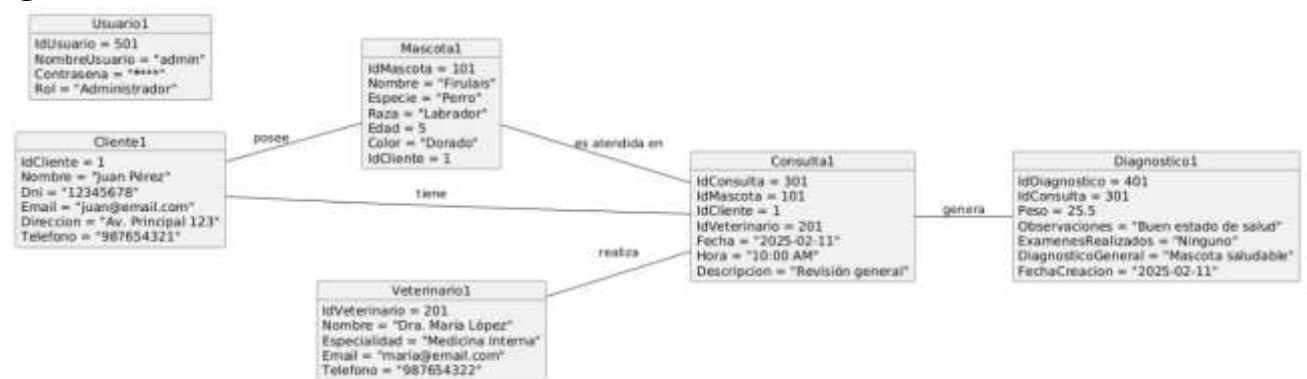
```
}

```

```
object Usuario1 {
  IdUsuario = 501
  NombreUsuario = "admin"
  Contraseña = "*****"
  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
```

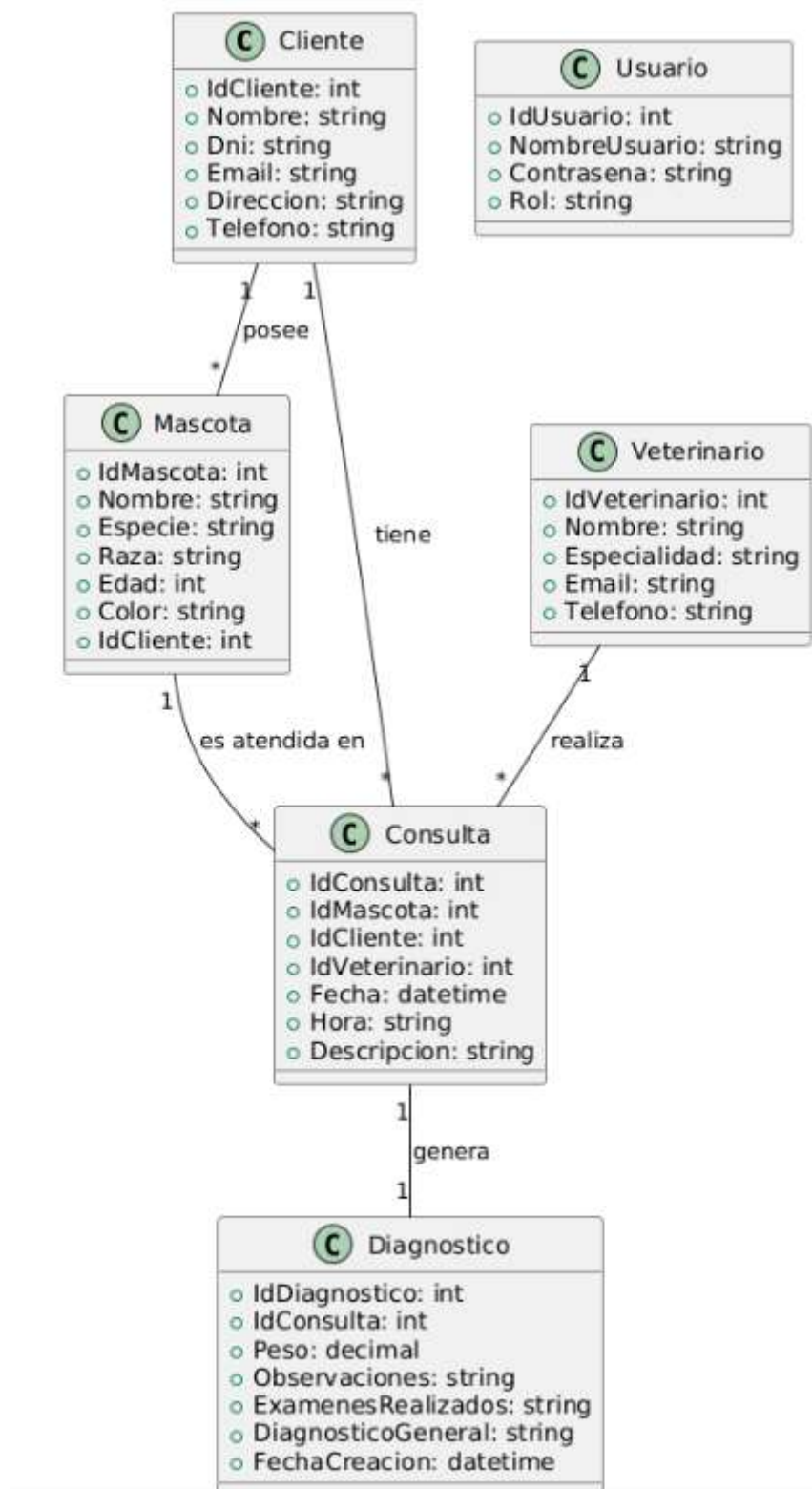


```
+Rol: string  
}
```

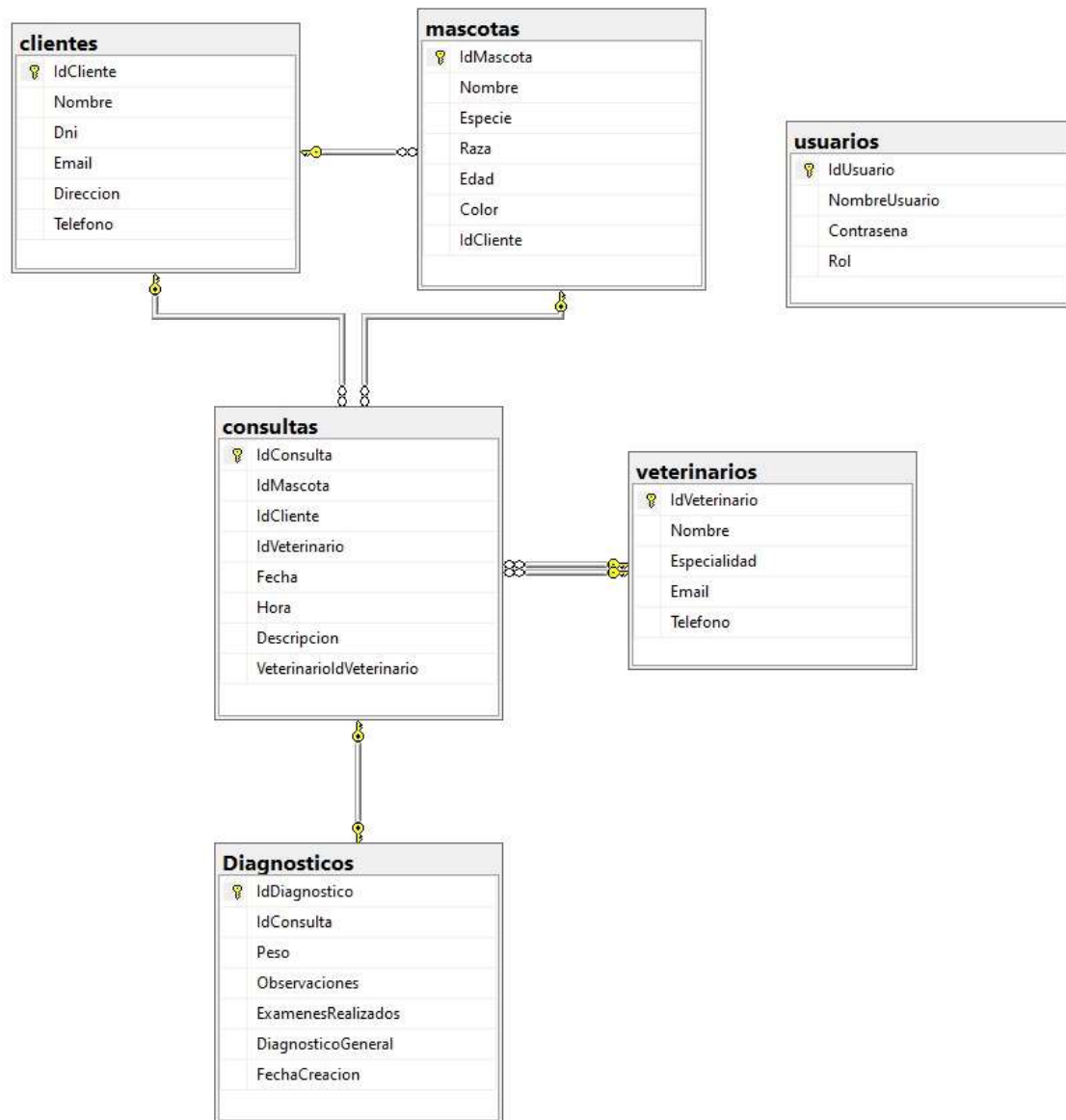
```
Cliente "1" -- "*" Mascota : posee  
Cliente "1" -- "*" Consulta : tiene  
Mascota "1" -- "*" Consulta : es atendida en  
Veterinario "1" -- "*" Consulta : realiza  
Consulta "1" -- "1" Diagnostico : genera
```

```
@enduml
```





### 3.2.6. Diagrama de Base de Datos



### 3.3. Vista de Implementación

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

```

graph TD;
    subgraph Frontend
        UI["Interfaz de Usuario (Web/App)"]
    end
    subgraph Backend
        API["API REST"]
        DB["Base de Datos SQL Server"]
    end
  
```

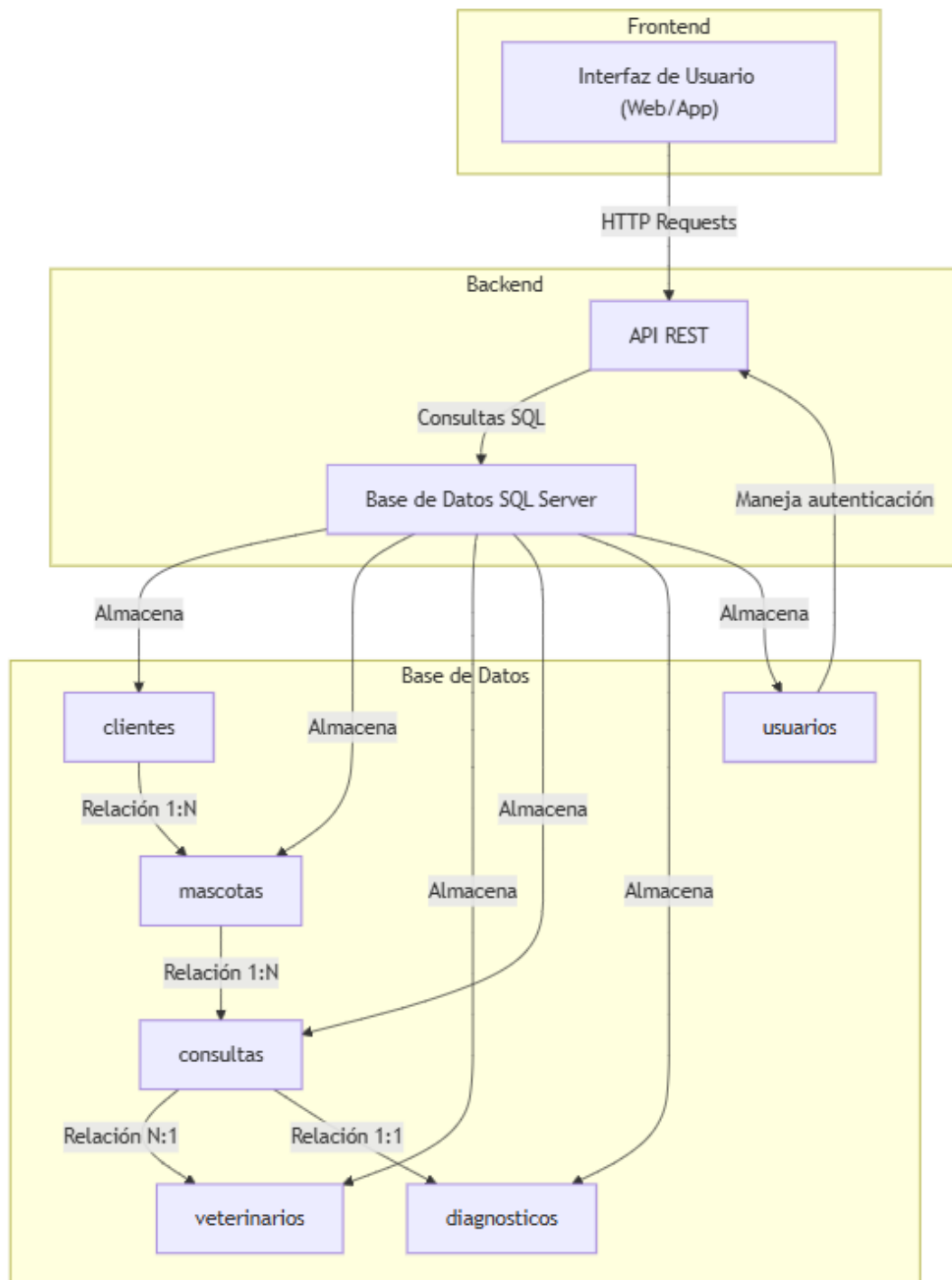


```
subgraph "Base de Datos"
  Clientes["clientes"]
  Mascotas["mascotas"]
  Veterinarios["veterinarios"]
  Consultas["consultas"]
  Diagnosticos["diagnosticos"]
  Usuarios["usuarios"]
end

UI -->|HTTP Requests| API
API -->|Consultas SQL| DB

DB -->|Almacena| Clientes
DB -->|Almacena| Mascotas
DB -->|Almacena| Veterinarios
DB -->|Almacena| Consultas
DB -->|Almacena| Diagnosticos
DB -->|Almacena| Usuarios

Clientes -->|Relación 1:N| Mascotas
Mascotas -->|Relación 1:N| Consultas
Consultas -->|Relación N:1| Veterinarios
Consultas -->|Relación 1:1| Diagnosticos
Usuarios -->|Maneja autenticación| API
```



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

```

graph TD;
    %% Definición de módulos principales
    subgraph "Frontend"
        UI["Interfaz de Usuario (Web/App)"]
    end
    end
  
```

```
subgraph "Backend"
    API["API REST"]
    Auth["Módulo de Autenticación"]
    Logic["Módulo de Lógica de Negocio"]
    Notificaciones["Módulo de Notificaciones"]
    Reportes["Módulo de Reportes PDF"]
end

subgraph "Base de Datos"
    DB["SQL Server"]
    Clientes["Tabla: clientes"]
    Mascotas["Tabla: mascotas"]
    Veterinarios["Tabla: veterinarios"]
    Consultas["Tabla: consultas"]
    Diagnosticos["Tabla: diagnosticos"]
    Usuarios["Tabla: usuarios"]
    Productos["Tabla: productos"]
    Ventas["Tabla: ventas"]
end

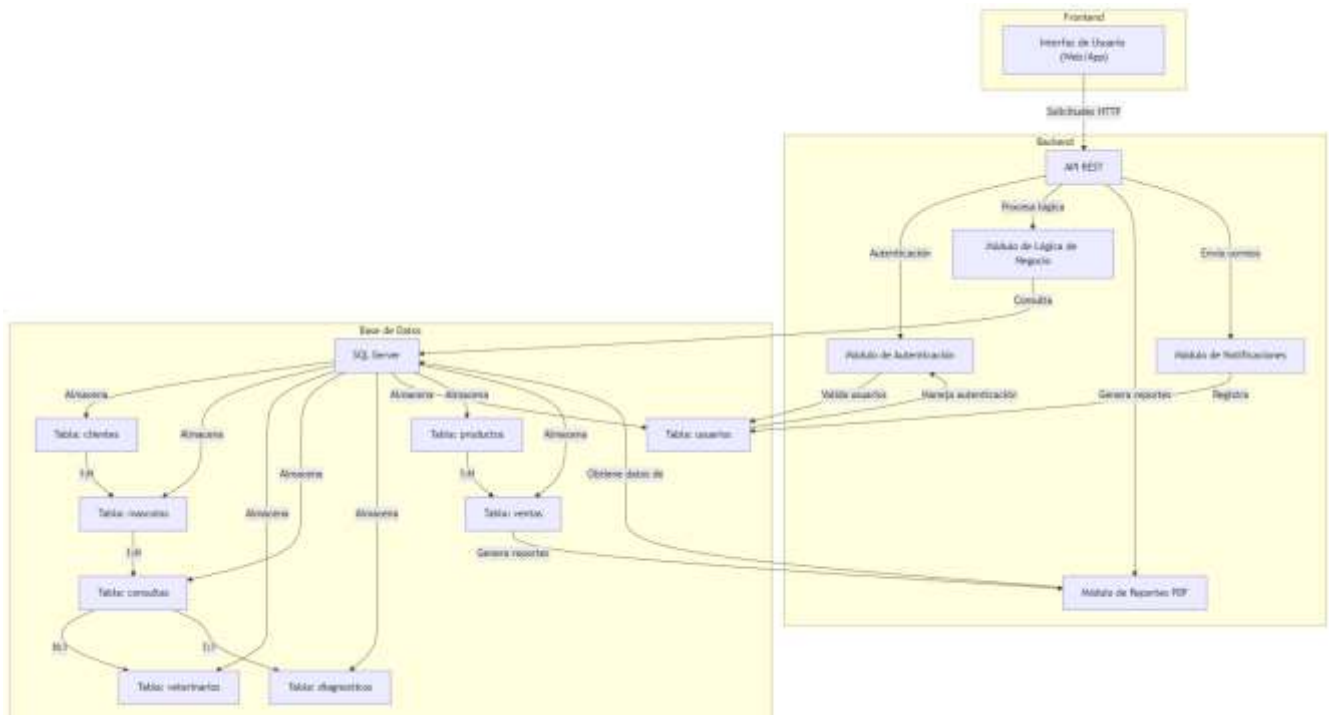
%% Conexiones entre módulos
UI -->|Solicitudes HTTP| API
API -->|Autenticación| Auth
API -->|Procesa lógica| Logic
API -->|Genera reportes| Reportes
API -->|Envía correos| Notificaciones

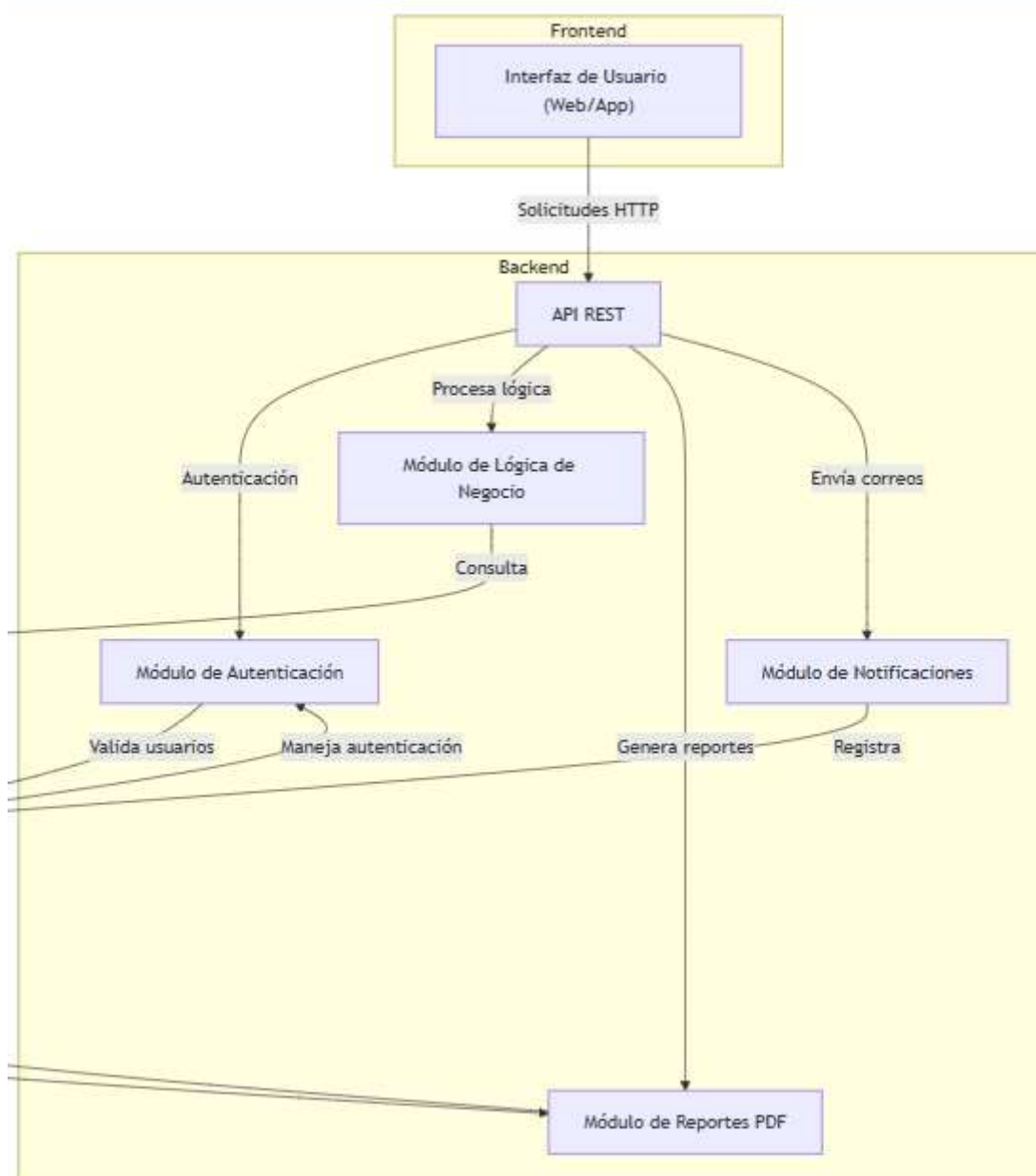
Logic -->|Consulta| DB
Auth -->|Valida usuarios| Usuarios
Notificaciones -->|Registra| Usuarios
Reportes -->|Obtiene datos de| DB

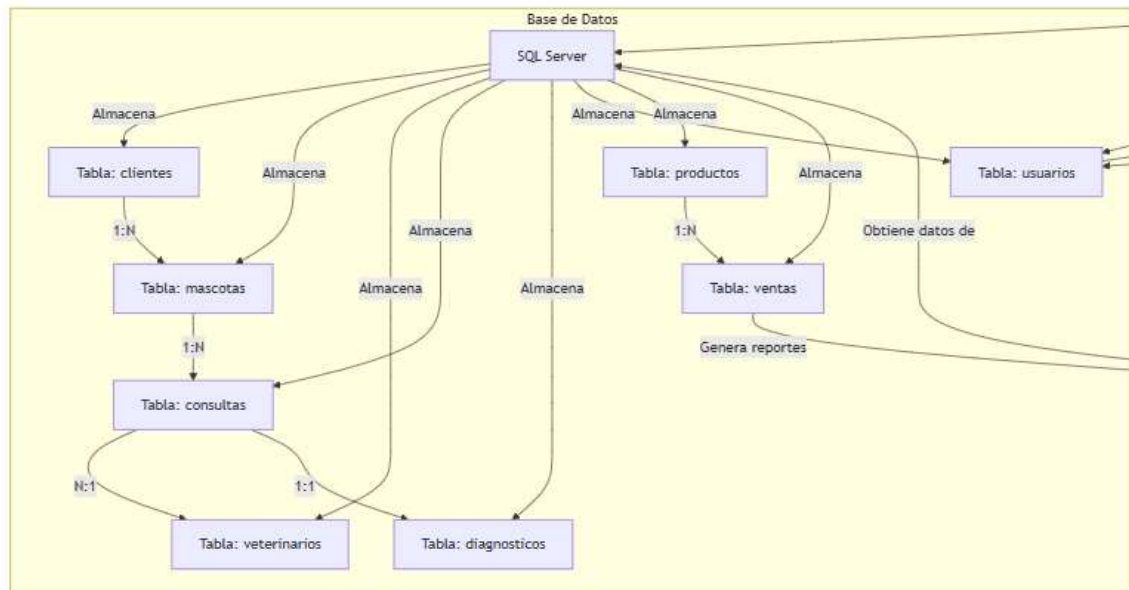
%% Conexiones de la base de datos
DB -->|Almacena| Clientes
DB -->|Almacena| Mascotas
DB -->|Almacena| Veterinarios
DB -->|Almacena| Consultas
DB -->|Almacena| Diagnosticos
DB -->|Almacena| Usuarios
DB -->|Almacena| Productos
DB -->|Almacena| Ventas

%% Relaciones entre tablas
Clientes -->|1:N| Mascotas
Mascotas -->|1:N| Consultas
Consultas -->|N:1| Veterinarios
```

Consultas -->|1:1| Diagnosticos  
Usuarios -->|Maneja autenticación| Auth  
Productos -->|1:N| Ventas  
Ventas -->|Genera reportes| Reportes







## 3.4. Vista de Procesos

### 3.4.1. Diagrama de Actividad

Mermaid:

graph TD;

```

%% Actores
Cliente["👤 Cliente"]
Veterinario["👨‍⚕️ Veterinario"]
Administrador["👤 Administrador"]
Sistema["💻 Sistema"]

%% Inicio
Start(["🚩 Inicio"]) -->|Cliente ingresa credenciales|
IniciarSesion["🔑 Iniciar Sesión"]
IniciarSesion --> ValidarCredenciales["🔑 Validar Credenciales"]

%% Validación de credenciales
ValidarCredenciales -->|✅ Válidas| AccederSistema["✅ Acceder al Sistema"]
ValidarCredenciales -->|❌ Inválidas| MostrarError["⚠️ Mostrar Error"]
MostrarError -->|Fin| End(["🚩 Fin"])

%% Cliente gestiona mascotas y consultas
AccederSistema -->|Registrar mascota| RegistrarMascota["👤 Registrar Mascota"]
  
```





```
RegistrarMascota --> GuardarMascotaBD["📄 Guardar en BD"]

AccederSistema -->|Solicitar consulta| SolicitarConsulta["📅 Solicitar Consulta"]
SolicitarConsulta --> RegistrarConsultaBD["📄 Registrar Consulta en BD"]

%% Veterinario revisa y atiende consultas
Veterinario -->|Ver consultas agendadas| VerConsultas["📄 Ver Consultas"]
VerConsultas --> MostrarDetalles["📄 Mostrar Detalles"]

Veterinario -->|Registrar diagnóstico| RegistrarDiagnostico["📝 Registrar Diagnóstico"]
RegistrarDiagnostico --> GuardarDiagnosticoBD["📄 Guardar en BD"]

%% Administrador gestiona productos y ventas
Administrador -->|Gestionar productos| GestionarProductos["📄 Gestionar Productos"]
GestionarProductos --> ActualizarProductosBD["📄 Actualizar BD"]

Administrador -->|Registrar venta| RegistrarVenta["💰 Registrar Venta"]
RegistrarVenta --> RegistrarVentaBD["📄 Registrar en BD"]

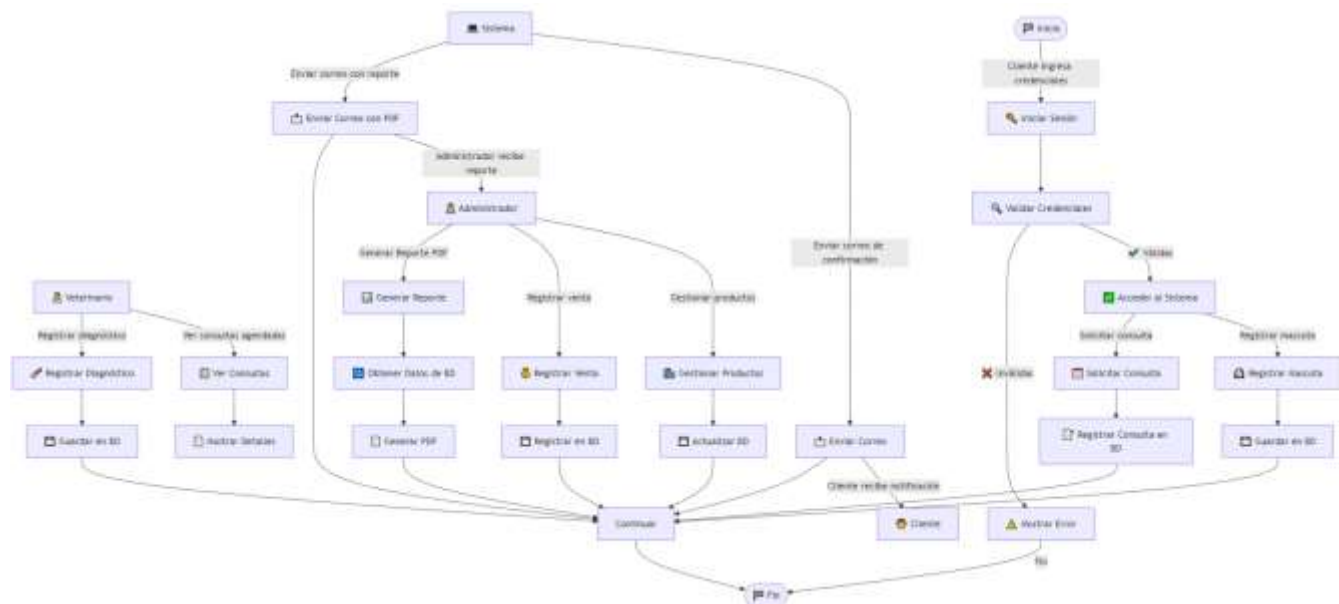
%% Generación de reportes en PDF
Administrador -->|Generar Reporte PDF| GenerarReporte["📊 Generar Reporte"]
GenerarReporte --> ObtenerDatosBD["📄 Obtener Datos de BD"]
ObtenerDatosBD --> GenerarPDF["📄 Generar PDF"]

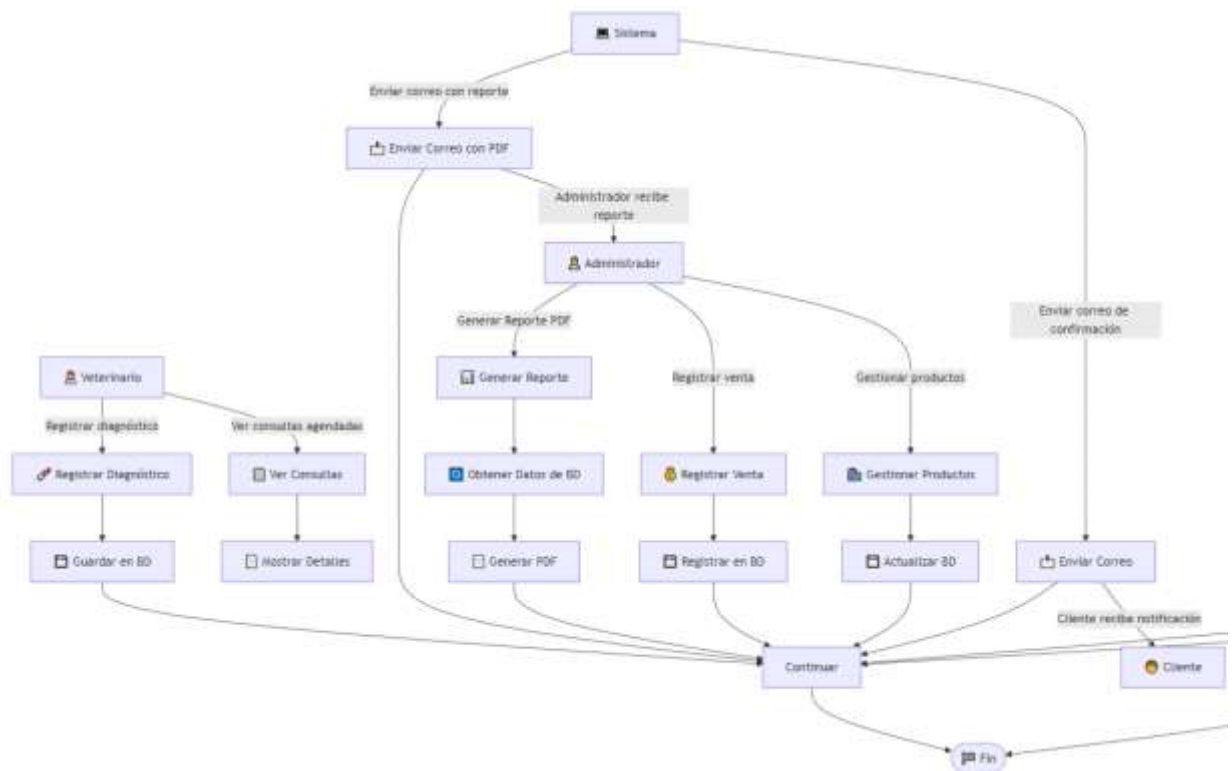
%% Notificaciones por correo
Sistema -->|Enviar correo de confirmación| EnviarCorreo["✉️ Enviar Correo"]
EnviarCorreo -->|Cliente recibe notificación| Cliente

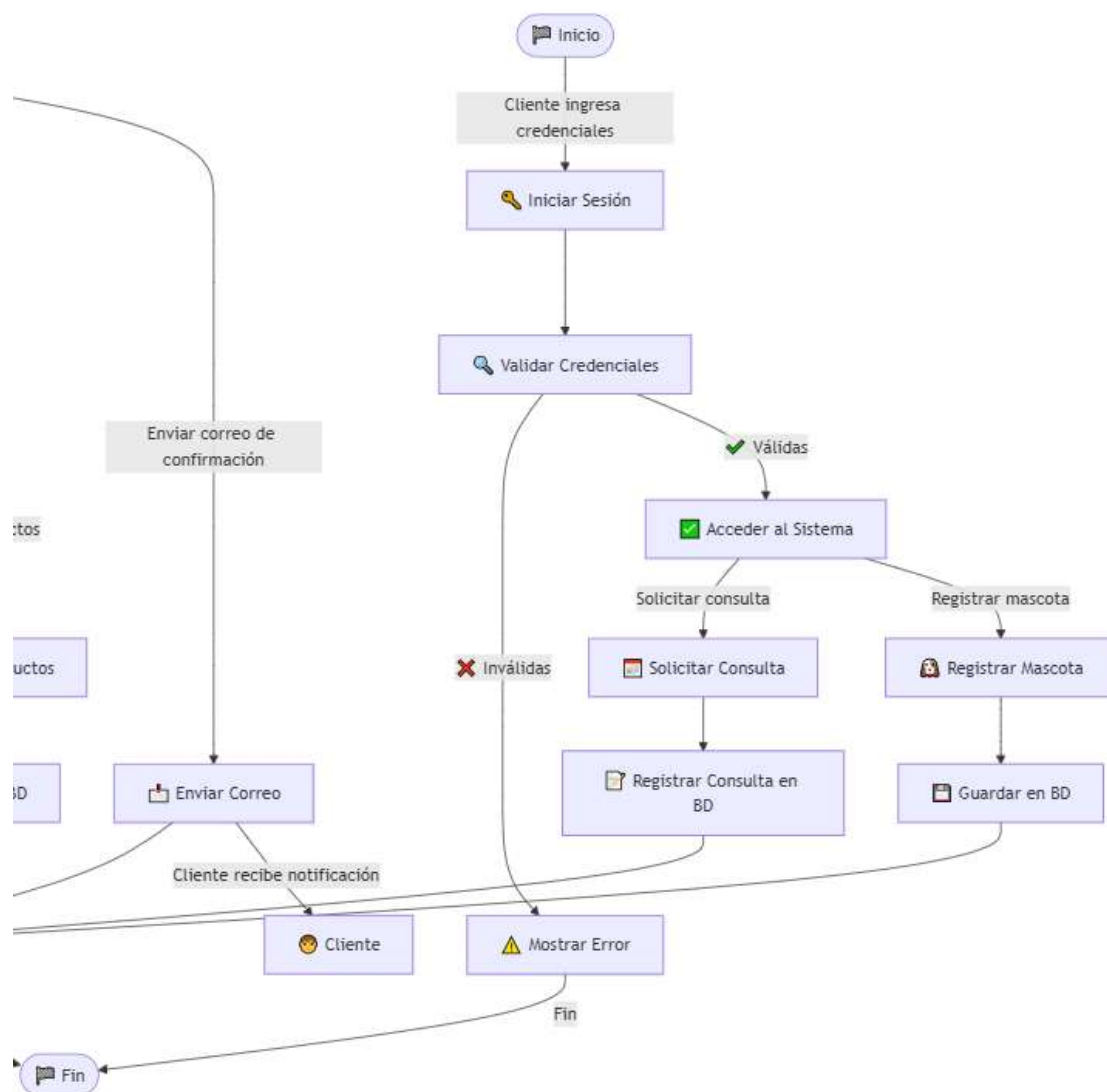
Sistema -->|Enviar correo con reporte| EnviarReporte["✉️ Enviar Correo con PDF"]
EnviarReporte -->|Administrador recibe reporte| Administrador

%% Fin del proceso
GuardarMascotaBD --> Continuar
RegistrarConsultaBD --> Continuar
GuardarDiagnosticoBD --> Continuar
ActualizarProductosBD --> Continuar
RegistrarVentaBD --> Continuar
GenerarPDF --> Continuar
```

EnviarCorreo --> Continuar  
EnviarReporte --> Continuar  
Continuar --> End







## 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.