

Andrés Alejandro Proaño Tapia

00326003

Proyecto Final

Base de Datos

Noel Pérez Pérez

Universidad San Francisco de Quito

* Problema

La base de datos mostrada en la imagen resuelve los problemas fundamentales de gestión en una clínica veterinaria, permitiendo el registro y seguimiento integral de pacientes animales, sus dueños y los servicios médicos prestados. A través de sus tablas principales y sus tablas de relación, el sistema facilita el control de citas médicas, el manejo de historiales clínicos, la administración de medicamentos, el seguimiento de tratamientos, y la gestión financiera mediante facturas. Además, las vistas especializadas permiten obtener información consolidada y detallada para la toma de decisiones, mejorando la eficiencia operativa de la clínica y la calidad de atención a las mascotas y sus propietarios.

* Cómo se resolvió el problema

Para resolver este problema de la veterinaria se crearon diferentes tablas que representan lo fundamental para la operación de una veterinaria:

* Dueños
* Mascotas
* Especies
* Facturas
* Citas
* Veterinarios
* Tratamientos
* Especialidades
* Medicamentos
* Implementación técnica

Backend (Node.js/Express): El servidor backend está construido con Express.js e incluye:

1. Core Configuration:

javascriptCopyconst express = require("express");

const app = express();

const mysql = require("mysql2");

const cors = require("cors");

1. Database Connection:

javascriptCopyconst db = mysql.createConnection({

host: "localhost",

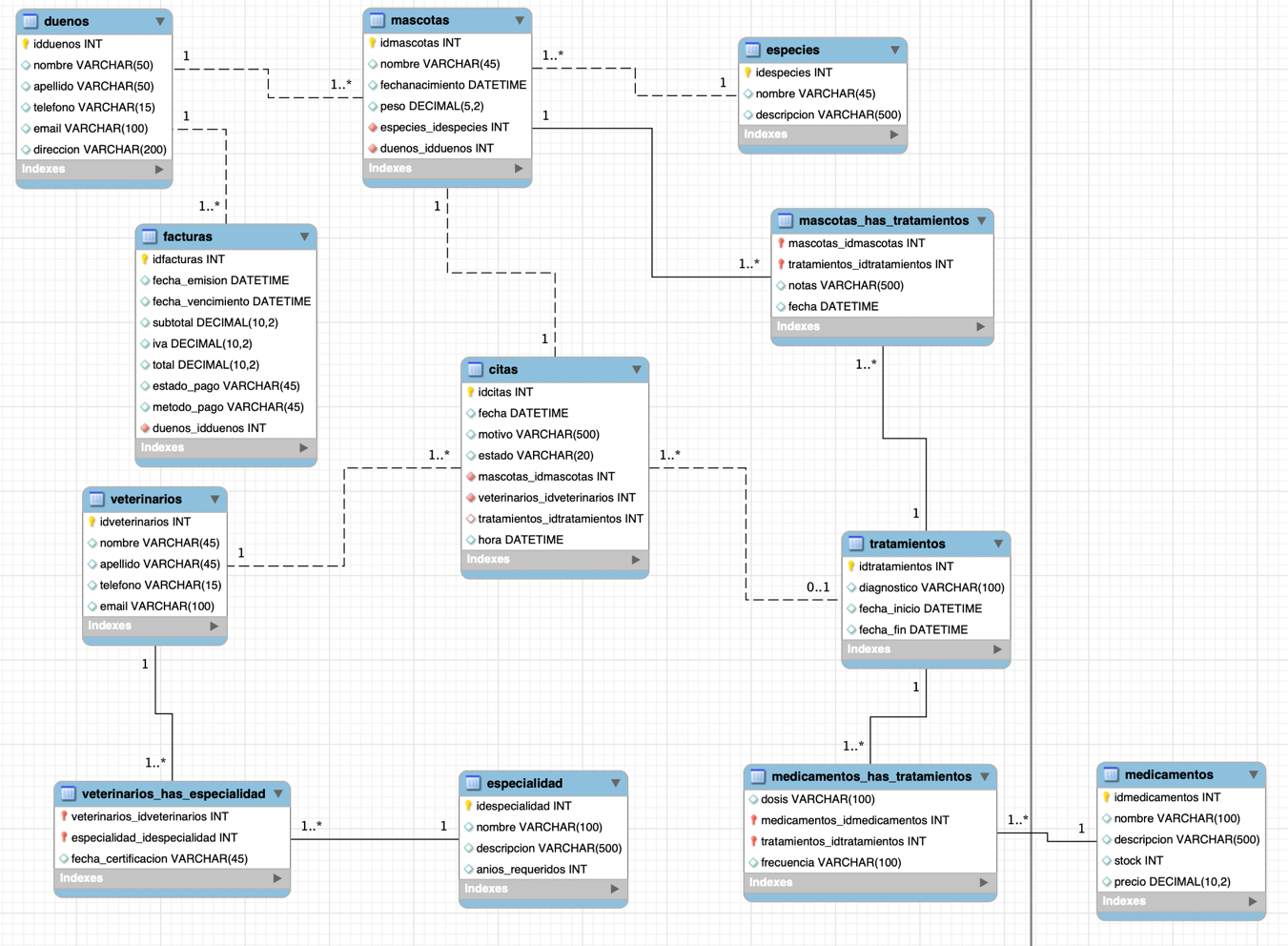
user: "root",

password: "\*\*\*\*",

database: "veterinaria"

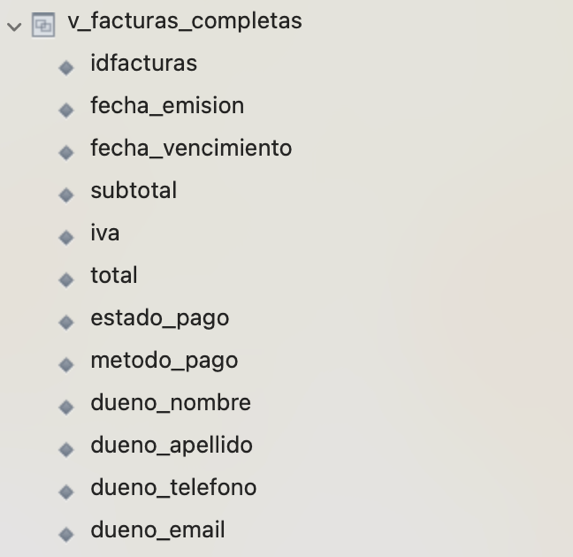
});

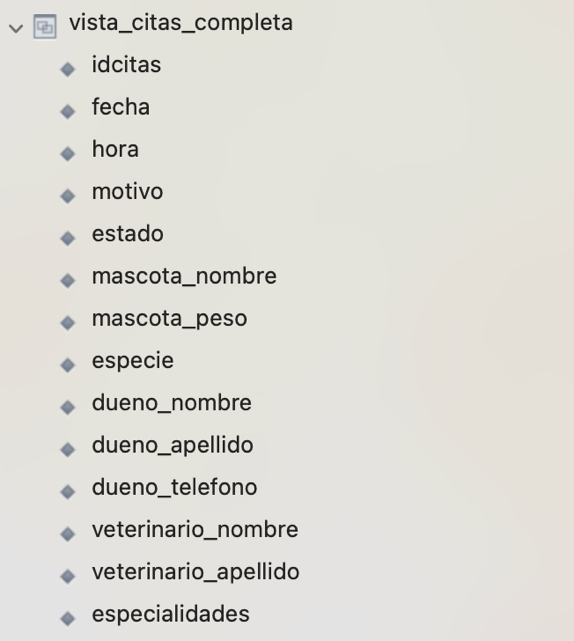
* + DER

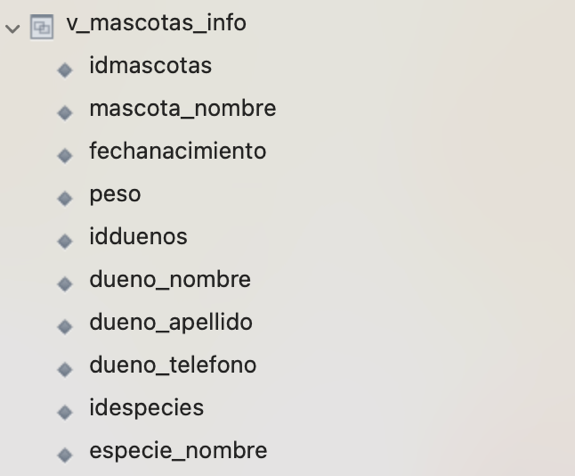


* + Views

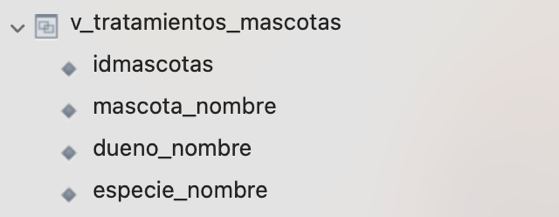
Se utilizaron distintas VIEWS para el mejor manejo de las consultas, en este caso se presentan las imágenes de las VIEWS:

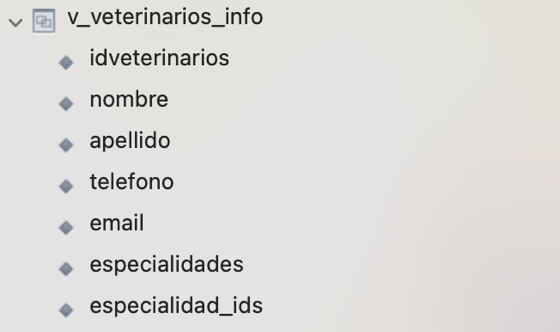












* + Consultas

Las consultas son las siguientes, que se crearon a través de procedures:

* + - Actualizar citas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_actualizar\_cita`(

IN p\_idcitas INT,

IN p\_fecha DATE,

IN p\_hora TIME,

IN p\_motivo VARCHAR(255),

IN p\_estado VARCHAR(50),

IN p\_mascotas\_idmascotas INT,

IN p\_veterinarios\_idveterinarios INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al actualizar la cita';

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que la cita exista

IF NOT EXISTS (SELECT 1 FROM citas WHERE idcitas = p\_idcitas) THEN

SET p\_resultado = 'La cita no existe';

ROLLBACK;

ELSE

-- Actualizar la cita

UPDATE citas

SET

fecha = p\_fecha,

hora = p\_hora,

motivo = p\_motivo,

estado = p\_estado,

mascotas\_idmascotas = p\_mascotas\_idmascotas,

veterinarios\_idveterinarios = p\_veterinarios\_idveterinarios

WHERE idcitas = p\_idcitas;

SET p\_resultado = 'Cita actualizada con éxito';

COMMIT;

END IF;

END

* + - Actualizar facturas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_actualizar\_factura`(

IN p\_idfacturas INT,

IN p\_fecha\_emision DATE,

IN p\_fecha\_vencimiento DATE,

IN p\_subtotal DECIMAL(10,2),

IN p\_iva DECIMAL(10,2),

IN p\_total DECIMAL(10,2),

IN p\_estado\_pago VARCHAR(50),

IN p\_metodo\_pago VARCHAR(50),

IN p\_duenos\_idduenos INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al actualizar la factura';

END;

START TRANSACTION;

-- Validar que exista la factura

IF NOT EXISTS (SELECT 1 FROM facturas WHERE idfacturas = p\_idfacturas) THEN

SET p\_resultado = 'La factura no existe';

ROLLBACK;

ELSE

-- Validar que exista el dueño

IF NOT EXISTS (SELECT 1 FROM duenos WHERE idduenos = p\_duenos\_idduenos) THEN

SET p\_resultado = 'El dueño especificado no existe';

ROLLBACK;

ELSE

-- Validar que la fecha de vencimiento sea posterior a la fecha de emisión

IF p\_fecha\_vencimiento < p\_fecha\_emision THEN

SET p\_resultado = 'La fecha de vencimiento debe ser posterior a la fecha de emisión';

ROLLBACK;

ELSE

-- Validar que el total coincida con subtotal + IVA

IF p\_total != (p\_subtotal + p\_iva) THEN

SET p\_resultado = 'El total no coincide con la suma de subtotal e IVA';

ROLLBACK;

ELSE

-- Actualizar la factura

UPDATE facturas

SET

fecha\_emision = p\_fecha\_emision,

fecha\_vencimiento = p\_fecha\_vencimiento,

subtotal = p\_subtotal,

iva = p\_iva,

total = p\_total,

estado\_pago = p\_estado\_pago,

metodo\_pago = p\_metodo\_pago,

duenos\_idduenos = p\_duenos\_idduenos

WHERE idfacturas = p\_idfacturas;

SET p\_resultado = 'Factura actualizada con éxito';

COMMIT;

END IF;

END IF;

END IF;

END IF;

END

* + - Actualizar mascotas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_actualizar\_mascota`(

IN p\_idmascotas INT,

IN p\_nombre VARCHAR(255),

IN p\_fechanacimiento DATE,

IN p\_peso DECIMAL(10,2),

IN p\_especies\_idespecies INT,

IN p\_duenos\_idduenos INT,

OUT p\_resultado VARCHAR(255),

OUT p\_filas\_afectadas INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al actualizar la mascota';

SET p\_filas\_afectadas = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que todos los campos requeridos tengan valor

IF p\_idmascotas IS NULL OR p\_nombre IS NULL OR p\_fechanacimiento IS NULL OR

p\_peso IS NULL OR p\_especies\_idespecies IS NULL OR p\_duenos\_idduenos IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_filas\_afectadas = 0;

ROLLBACK;

ELSE

-- Validar que exista la mascota

IF NOT EXISTS (SELECT 1 FROM mascotas WHERE idmascotas = p\_idmascotas) THEN

SET p\_resultado = 'La mascota no existe';

SET p\_filas\_afectadas = 0;

ROLLBACK;

-- Validar que exista la especie

ELSEIF NOT EXISTS (SELECT 1 FROM especies WHERE idespecies = p\_especies\_idespecies) THEN

SET p\_resultado = 'La especie especificada no existe';

SET p\_filas\_afectadas = 0;

ROLLBACK;

-- Validar que exista el dueño

ELSEIF NOT EXISTS (SELECT 1 FROM duenos WHERE idduenos = p\_duenos\_idduenos) THEN

SET p\_resultado = 'El dueño especificado no existe';

SET p\_filas\_afectadas = 0;

ROLLBACK;

ELSE

-- Actualizar la mascota

UPDATE mascotas

SET

nombre = p\_nombre,

fechanacimiento = p\_fechanacimiento,

peso = p\_peso,

especies\_idespecies = p\_especies\_idespecies,

duenos\_idduenos = p\_duenos\_idduenos

WHERE idmascotas = p\_idmascotas;

SET p\_filas\_afectadas = ROW\_COUNT();

SET p\_resultado = 'Mascota actualizada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Actualizar tratamientos

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_actualizar\_tratamiento`(

IN p\_idtratamientos INT,

IN p\_diagnostico TEXT,

IN p\_fecha\_inicio DATE,

IN p\_fecha\_fin DATE,

IN p\_mascota\_id INT,

IN p\_veterinario\_id INT,

IN p\_motivo\_cita TEXT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE v\_cita\_existe INT DEFAULT 0;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al actualizar el tratamiento';

END;

START TRANSACTION;

-- Validar campos requeridos

IF p\_idtratamientos IS NULL OR p\_diagnostico IS NULL OR

p\_fecha\_inicio IS NULL OR p\_mascota\_id IS NULL OR

p\_veterinario\_id IS NULL OR p\_motivo\_cita IS NULL THEN

SET p\_resultado = 'Todos los campos requeridos son necesarios';

ROLLBACK;

ELSE

-- Validar que exista el tratamiento

IF NOT EXISTS (SELECT 1 FROM tratamientos WHERE idtratamientos = p\_idtratamientos) THEN

SET p\_resultado = 'El tratamiento no existe';

ROLLBACK;

-- Validar que exista la mascota

ELSEIF NOT EXISTS (SELECT 1 FROM mascotas WHERE idmascotas = p\_mascota\_id) THEN

SET p\_resultado = 'La mascota especificada no existe';

ROLLBACK;

-- Validar que exista el veterinario

ELSEIF NOT EXISTS (SELECT 1 FROM veterinarios WHERE idveterinarios = p\_veterinario\_id) THEN

SET p\_resultado = 'El veterinario especificado no existe';

ROLLBACK;

ELSE

-- Actualizar tratamiento

UPDATE tratamientos

SET

diagnostico = p\_diagnostico,

fecha\_inicio = p\_fecha\_inicio,

fecha\_fin = p\_fecha\_fin

WHERE idtratamientos = p\_idtratamientos;

-- Actualizar relación mascota-tratamiento

UPDATE mascotas\_has\_tratamientos

SET mascotas\_idmascotas = p\_mascota\_id

WHERE tratamientos\_idtratamientos = p\_idtratamientos;

-- Verificar si existe una cita asociada

SELECT COUNT(\*) INTO v\_cita\_existe

FROM citas

WHERE tratamientos\_idtratamientos = p\_idtratamientos;

IF v\_cita\_existe > 0 THEN

-- Actualizar cita existente

UPDATE citas

SET

fecha = p\_fecha\_inicio,

motivo = p\_motivo\_cita,

mascotas\_idmascotas = p\_mascota\_id,

veterinarios\_idveterinarios = p\_veterinario\_id

WHERE tratamientos\_idtratamientos = p\_idtratamientos;

ELSE

-- Crear nueva cita

INSERT INTO citas (

fecha,

motivo,

mascotas\_idmascotas,

veterinarios\_idveterinarios,

tratamientos\_idtratamientos

) VALUES (

p\_fecha\_inicio,

p\_motivo\_cita,

p\_mascota\_id,

p\_veterinario\_id,

p\_idtratamientos

);

END IF;

SET p\_resultado = 'Tratamiento y cita actualizados con éxito';

COMMIT;

END IF;

END IF;

END

* + - Actualizar veterinarios

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_actualizar\_veterinario`(

IN p\_idveterinarios INT,

IN p\_nombre VARCHAR(255),

IN p\_apellido VARCHAR(255),

IN p\_telefono VARCHAR(20),

IN p\_email VARCHAR(255),

IN p\_especialidad\_ids INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE v\_existing\_email VARCHAR(255);

DECLARE v\_existing\_phone VARCHAR(20);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al actualizar el veterinario';

END;

START TRANSACTION;

-- Validar que exista el veterinario

IF NOT EXISTS (SELECT 1 FROM veterinarios WHERE idveterinarios = p\_idveterinarios) THEN

SET p\_resultado = 'El veterinario no existe';

ROLLBACK;

ELSE

-- Verificar duplicados de email y teléfono (excluyendo el registro actual)

SELECT email, telefono INTO v\_existing\_email, v\_existing\_phone

FROM veterinarios

WHERE (email = p\_email OR telefono = p\_telefono)

AND idveterinarios != p\_idveterinarios

LIMIT 1;

IF v\_existing\_email IS NOT NULL THEN

SET p\_resultado = 'Ya existe otro veterinario con este email';

ROLLBACK;

ELSEIF v\_existing\_phone IS NOT NULL THEN

SET p\_resultado = 'Ya existe otro veterinario con este teléfono';

ROLLBACK;

ELSE

-- Validar que exista la especialidad

IF NOT EXISTS (SELECT 1 FROM especialidad WHERE idespecialidad = p\_especialidad\_ids) THEN

SET p\_resultado = 'La especialidad especificada no existe';

ROLLBACK;

ELSE

-- Actualizar información básica

UPDATE veterinarios

SET

nombre = p\_nombre,

apellido = p\_apellido,

telefono = p\_telefono,

email = p\_email

WHERE idveterinarios = p\_idveterinarios;

-- Actualizar especialidades

DELETE FROM veterinarios\_has\_especialidad

WHERE veterinarios\_idveterinarios = p\_idveterinarios;

INSERT INTO veterinarios\_has\_especialidad

(veterinarios\_idveterinarios, especialidad\_idespecialidad, fecha\_certificacion)

VALUES (p\_idveterinarios, p\_especialidad\_ids, CURRENT\_DATE);

SET p\_resultado = 'Veterinario y especialidad actualizados con éxito';

COMMIT;

END IF;

END IF;

END IF;

END

* + - Crear cita

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_cita`(

IN p\_fecha DATE,

IN p\_hora TIME,

IN p\_motivo VARCHAR(255),

IN p\_estado VARCHAR(50),

IN p\_mascotas\_idmascotas INT,

IN p\_veterinarios\_idveterinarios INT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar la cita';

SET p\_id\_generado = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que todos los campos requeridos tengan valor

IF p\_fecha IS NULL OR p\_hora IS NULL OR p\_motivo IS NULL OR

p\_estado IS NULL OR p\_mascotas\_idmascotas IS NULL OR

p\_veterinarios\_idveterinarios IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar la cita

INSERT INTO citas(

idcitas,

fecha,

hora,

motivo,

estado,

mascotas\_idmascotas,

veterinarios\_idveterinarios

) VALUES (

DEFAULT,

p\_fecha,

p\_hora,

p\_motivo,

p\_estado,

p\_mascotas\_idmascotas,

p\_veterinarios\_idveterinarios

);

-- Obtener el ID generado

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Cita Registrada con éxito';

COMMIT;

END IF;

END

* + - Crear dueno

REATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_dueno`(

IN p\_nombre VARCHAR(255),

IN p\_apellido VARCHAR(255),

IN p\_telefono VARCHAR(20),

IN p\_email VARCHAR(255),

IN p\_direccion TEXT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar el dueño';

SET p\_id\_generado = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que los campos requeridos tengan valor

IF p\_nombre IS NULL OR p\_apellido IS NULL OR p\_telefono IS NULL THEN

SET p\_resultado = 'Nombre, apellido y teléfono son campos requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar si ya existe un dueño con el mismo teléfono

IF EXISTS (SELECT 1 FROM duenos WHERE telefono = p\_telefono) THEN

SET p\_resultado = 'Ya existe un dueño registrado con este teléfono';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar si ya existe un dueño con el mismo email (si se proporcionó)

IF p\_email IS NOT NULL AND EXISTS (SELECT 1 FROM duenos WHERE email = p\_email) THEN

SET p\_resultado = 'Ya existe un dueño registrado con este email';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar el dueño

INSERT INTO duenos(

idduenos,

nombre,

apellido,

telefono,

email,

direccion

) VALUES (

DEFAULT,

p\_nombre,

p\_apellido,

p\_telefono,

p\_email,

p\_direccion

);

-- Obtener el ID generado

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Dueño registrado con éxito';

COMMIT;

END IF;

END IF;

END IF;

END

* + - Crear especialidad

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_especialidad`(

IN p\_nombre VARCHAR(255),

IN p\_descripcion TEXT,

IN p\_anios\_requeridos INT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar la especialidad';

SET p\_id\_generado = 0;

END;

START TRANSACTION;

-- Validar campos requeridos

IF p\_nombre IS NULL OR p\_descripcion IS NULL OR p\_anios\_requeridos IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que los años requeridos sean un número positivo

IF p\_anios\_requeridos <= 0 THEN

SET p\_resultado = 'Los años requeridos deben ser un número positivo';

SET p\_id\_generado = 0;

ROLLBACK;

-- Validar que no exista otra especialidad con el mismo nombre

ELSEIF EXISTS (SELECT 1 FROM especialidad WHERE LOWER(nombre) = LOWER(p\_nombre)) THEN

SET p\_resultado = 'Ya existe una especialidad con este nombre';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar la especialidad

INSERT INTO especialidad(

idespecialidad,

nombre,

descripcion,

anios\_requeridos

) VALUES (

DEFAULT,

p\_nombre,

p\_descripcion,

p\_anios\_requeridos

);

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Especialidad registrada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Crear especie

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_especie`(

IN p\_nombre VARCHAR(255),

IN p\_descripcion TEXT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar la especie';

SET p\_id\_generado = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que todos los campos requeridos tengan valor

IF p\_nombre IS NULL OR p\_descripcion IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar si ya existe una especie con el mismo nombre

IF EXISTS (SELECT 1 FROM especies WHERE nombre = p\_nombre) THEN

SET p\_resultado = 'Ya existe una especie con este nombre';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar la especie

INSERT INTO especies(

idespecies,

nombre,

descripcion

) VALUES (

DEFAULT,

p\_nombre,

p\_descripcion

);

-- Obtener el ID generado

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Especie Registrada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Crear factura

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_factura`(

IN p\_fecha\_emision DATE,

IN p\_fecha\_vencimiento DATE,

IN p\_subtotal DECIMAL(10,2),

IN p\_iva DECIMAL(10,2),

IN p\_total DECIMAL(10,2),

IN p\_estado\_pago VARCHAR(50),

IN p\_metodo\_pago VARCHAR(50),

IN p\_duenos\_idduenos INT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar la factura';

SET p\_id\_generado = 0;

END;

START TRANSACTION;

-- Validar campos requeridos

IF p\_fecha\_emision IS NULL OR p\_fecha\_vencimiento IS NULL OR

p\_subtotal IS NULL OR p\_iva IS NULL OR p\_total IS NULL OR

p\_estado\_pago IS NULL OR p\_metodo\_pago IS NULL OR

p\_duenos\_idduenos IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que exista el dueño

IF NOT EXISTS (SELECT 1 FROM duenos WHERE idduenos = p\_duenos\_idduenos) THEN

SET p\_resultado = 'El dueño especificado no existe';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que la fecha de vencimiento sea posterior a la fecha de emisión

IF p\_fecha\_vencimiento < p\_fecha\_emision THEN

SET p\_resultado = 'La fecha de vencimiento debe ser posterior a la fecha de emisión';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que el total coincida con subtotal + IVA

IF p\_total != (p\_subtotal + p\_iva) THEN

SET p\_resultado = 'El total no coincide con la suma de subtotal e IVA';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar la factura

INSERT INTO facturas(

fecha\_emision,

fecha\_vencimiento,

subtotal,

iva,

total,

estado\_pago,

metodo\_pago,

duenos\_idduenos

) VALUES (

p\_fecha\_emision,

p\_fecha\_vencimiento,

p\_subtotal,

p\_iva,

p\_total,

p\_estado\_pago,

p\_metodo\_pago,

p\_duenos\_idduenos

);

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Factura registrada con éxito';

COMMIT;

END IF;

END IF;

END IF;

END IF;

END

* + - Crear mascota

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_mascota`(

IN p\_nombre VARCHAR(255),

IN p\_fechanacimiento DATE,

IN p\_peso DECIMAL(10,2),

IN p\_especies\_idespecies INT,

IN p\_duenos\_idduenos INT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar la mascota';

SET p\_id\_generado = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que todos los campos requeridos tengan valor

IF p\_nombre IS NULL OR p\_fechanacimiento IS NULL OR

p\_peso IS NULL OR p\_especies\_idespecies IS NULL OR

p\_duenos\_idduenos IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que exista la especie y el dueño

IF NOT EXISTS (SELECT 1 FROM especies WHERE idespecies = p\_especies\_idespecies) THEN

SET p\_resultado = 'La especie especificada no existe';

SET p\_id\_generado = 0;

ROLLBACK;

ELSEIF NOT EXISTS (SELECT 1 FROM duenos WHERE idduenos = p\_duenos\_idduenos) THEN

SET p\_resultado = 'El dueño especificado no existe';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar la mascota

INSERT INTO mascotas(

idmascotas,

nombre,

fechanacimiento,

peso,

especies\_idespecies,

duenos\_idduenos

) VALUES (

DEFAULT,

p\_nombre,

p\_fechanacimiento,

p\_peso,

p\_especies\_idespecies,

p\_duenos\_idduenos

);

-- Obtener el ID generado

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Mascota Registrada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Crear tratamiento

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_tratamiento`(

IN p\_diagnostico TEXT,

IN p\_fecha\_inicio DATE,

IN p\_fecha\_fin DATE,

IN p\_mascota\_id INT,

IN p\_veterinario\_id INT,

IN p\_motivo\_cita TEXT,

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE v\_tratamiento\_id INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar el tratamiento';

SET p\_id\_generado = 0;

END;

START TRANSACTION;

-- Validar campos requeridos

IF p\_diagnostico IS NULL OR p\_fecha\_inicio IS NULL OR

p\_mascota\_id IS NULL OR p\_veterinario\_id IS NULL OR

p\_motivo\_cita IS NULL THEN

SET p\_resultado = 'Campos requeridos faltantes';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar que exista la mascota

IF NOT EXISTS (SELECT 1 FROM mascotas WHERE idmascotas = p\_mascota\_id) THEN

SET p\_resultado = 'La mascota especificada no existe';

SET p\_id\_generado = 0;

ROLLBACK;

-- Validar que exista el veterinario

ELSEIF NOT EXISTS (SELECT 1 FROM veterinarios WHERE idveterinarios = p\_veterinario\_id) THEN

SET p\_resultado = 'El veterinario especificado no existe';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar el tratamiento

INSERT INTO tratamientos (

diagnostico,

fecha\_inicio,

fecha\_fin

) VALUES (

p\_diagnostico,

p\_fecha\_inicio,

p\_fecha\_fin

);

SET v\_tratamiento\_id = LAST\_INSERT\_ID();

-- Crear la relación mascota-tratamiento

INSERT INTO mascotas\_has\_tratamientos (

mascotas\_idmascotas,

tratamientos\_idtratamientos

) VALUES (

p\_mascota\_id,

v\_tratamiento\_id

);

-- Crear la cita asociada

INSERT INTO citas (

fecha,

motivo,

mascotas\_idmascotas,

veterinarios\_idveterinarios,

tratamientos\_idtratamientos

) VALUES (

p\_fecha\_inicio,

p\_motivo\_cita,

p\_mascota\_id,

p\_veterinario\_id,

v\_tratamiento\_id

);

SET p\_id\_generado = v\_tratamiento\_id;

SET p\_resultado = 'Tratamiento registrado con éxito';

COMMIT;

END IF;

END IF;

END

* + - Crear veterinaria

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_veterinario`(

IN p\_nombre VARCHAR(255),

IN p\_apellido VARCHAR(255),

IN p\_telefono VARCHAR(20),

IN p\_email VARCHAR(255),

OUT p\_resultado VARCHAR(255),

OUT p\_id\_generado INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al registrar el veterinario';

SET p\_id\_generado = 0;

END;

START TRANSACTION;

-- Validar campos requeridos

IF p\_nombre IS NULL OR p\_apellido IS NULL OR p\_telefono IS NULL OR p\_email IS NULL THEN

SET p\_resultado = 'Todos los campos son requeridos';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Validar si ya existe un veterinario con el mismo email o teléfono

IF EXISTS (SELECT 1 FROM veterinarios WHERE email = p\_email) THEN

SET p\_resultado = 'Ya existe un veterinario con este email';

SET p\_id\_generado = 0;

ROLLBACK;

ELSEIF EXISTS (SELECT 1 FROM veterinarios WHERE telefono = p\_telefono) THEN

SET p\_resultado = 'Ya existe un veterinario con este teléfono';

SET p\_id\_generado = 0;

ROLLBACK;

ELSE

-- Insertar el veterinario

INSERT INTO veterinarios(

idveterinarios,

nombre,

apellido,

telefono,

email

) VALUES (

DEFAULT,

p\_nombre,

p\_apellido,

p\_telefono,

p\_email

);

SET p\_id\_generado = LAST\_INSERT\_ID();

SET p\_resultado = 'Veterinario registrado con éxito';

COMMIT;

END IF;

END IF;

END

* + - Eliminar cita

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_eliminar\_cita`(

IN p\_idcitas INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al eliminar la cita';

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que la cita exista

IF NOT EXISTS (SELECT 1 FROM citas WHERE idcitas = p\_idcitas) THEN

SET p\_resultado = 'La cita no existe';

ROLLBACK;

ELSE

-- Eliminar la cita

DELETE FROM citas

WHERE idcitas = p\_idcitas;

SET p\_resultado = 'Cita eliminada con éxito';

COMMIT;

END IF;

END

* + - Eliminar factura

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_eliminar\_factura`(

IN p\_idfacturas INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE v\_estado\_pago VARCHAR(50);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al eliminar la factura';

END;

START TRANSACTION;

-- Validar que exista la factura

IF NOT EXISTS (SELECT 1 FROM facturas WHERE idfacturas = p\_idfacturas) THEN

SET p\_resultado = 'La factura no existe';

ROLLBACK;

ELSE

-- Verificar el estado de pago

SELECT estado\_pago INTO v\_estado\_pago

FROM facturas

WHERE idfacturas = p\_idfacturas;

-- Solo permitir eliminar facturas no pagadas

IF v\_estado\_pago = 'PAGADO' THEN

SET p\_resultado = 'No se puede eliminar una factura pagada';

ROLLBACK;

ELSE

-- Eliminar la factura

DELETE FROM facturas

WHERE idfacturas = p\_idfacturas;

SET p\_resultado = 'Factura eliminada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Eliminar mascota

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_eliminar\_mascota`(

IN p\_idmascotas INT,

OUT p\_resultado VARCHAR(255),

OUT p\_filas\_afectadas INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al eliminar la mascota';

SET p\_filas\_afectadas = 0;

END;

-- Iniciar transacción

START TRANSACTION;

-- Validar que la mascota exista

IF NOT EXISTS (SELECT 1 FROM mascotas WHERE idmascotas = p\_idmascotas) THEN

SET p\_resultado = 'La mascota no existe';

SET p\_filas\_afectadas = 0;

ROLLBACK;

ELSE

-- Verificar si la mascota tiene citas asociadas

IF EXISTS (SELECT 1 FROM citas WHERE mascotas\_idmascotas = p\_idmascotas) THEN

SET p\_resultado = 'No se puede eliminar la mascota porque tiene citas asociadas';

SET p\_filas\_afectadas = 0;

ROLLBACK;

ELSE

-- Eliminar la mascota

DELETE FROM mascotas

WHERE idmascotas = p\_idmascotas;

SET p\_filas\_afectadas = ROW\_COUNT();

SET p\_resultado = 'Mascota eliminada con éxito';

COMMIT;

END IF;

END IF;

END

* + - Eliminar tratamiento

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_eliminar\_tratamiento`(

IN p\_idtratamientos INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al eliminar el tratamiento';

END;

START TRANSACTION;

-- Validar que exista el tratamiento

IF NOT EXISTS (SELECT 1 FROM tratamientos WHERE idtratamientos = p\_idtratamientos) THEN

SET p\_resultado = 'El tratamiento no existe';

ROLLBACK;

ELSE

-- Eliminar registros relacionados en orden

-- 1. Eliminar citas relacionadas

DELETE FROM citas

WHERE tratamientos\_idtratamientos = p\_idtratamientos;

-- 2. Eliminar relaciones mascota-tratamiento

DELETE FROM mascotas\_has\_tratamientos

WHERE tratamientos\_idtratamientos = p\_idtratamientos;

-- 3. Eliminar el tratamiento

DELETE FROM tratamientos

WHERE idtratamientos = p\_idtratamientos;

SET p\_resultado = 'Tratamiento y registros relacionados eliminados con éxito';

COMMIT;

END IF;

END

* + - Eliminar veterinario

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_eliminar\_veterinario`(

IN p\_idveterinarios INT,

OUT p\_resultado VARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SET p\_resultado = 'Error al eliminar el veterinario';

END;

START TRANSACTION;

-- Validar que exista el veterinario

IF NOT EXISTS (SELECT 1 FROM veterinarios WHERE idveterinarios = p\_idveterinarios) THEN

SET p\_resultado = 'El veterinario no existe';

ROLLBACK;

ELSE

-- Verificar si tiene citas asociadas

IF EXISTS (SELECT 1 FROM citas WHERE veterinarios\_idveterinarios = p\_idveterinarios) THEN

SET p\_resultado = 'No se puede eliminar el veterinario porque tiene citas asociadas';

ROLLBACK;

ELSE

-- Primero eliminar las relaciones con especialidades

DELETE FROM veterinarios\_has\_especialidad

WHERE veterinarios\_idveterinarios = p\_idveterinarios;

-- Luego eliminar el veterinario

DELETE FROM veterinarios

WHERE idveterinarios = p\_idveterinarios;

SET p\_resultado = 'Veterinario eliminado con éxito';

COMMIT;

END IF;

END IF;

END

* + - Obtener citas completas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_citas\_completas`()

BEGIN

SELECT

v.\*,

c.mascotas\_idmascotas,

c.veterinarios\_idveterinarios

FROM vista\_citas\_completa v

JOIN citas c ON v.idcitas = c.idcitas

ORDER BY v.idcitas;

END

* + - Obtener duenos

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_duenos`()

BEGIN

SELECT \*

FROM duenos

ORDER BY idduenos;

END

* + - Obtener especialidades

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_especialidades`()

BEGIN

SELECT \*

FROM especialidad;

END

* + - Obtener especies

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_especies`()

BEGIN

SELECT \*

FROM especies

ORDER BY idespecies;

END

* + - Obtener facturas info

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_facturas\_info`()

BEGIN

SELECT \*

FROM v\_facturas\_completas

ORDER BY idfacturas;

END

* + - Obtener mascotas info

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_mascota\_info`(

IN p\_idmascota INT,

OUT p\_encontrado BOOLEAN

)

BEGIN

DECLARE v\_count INT;

-- Verificar si existe la mascota

SELECT COUNT(\*) INTO v\_count

FROM v\_mascotas\_info

WHERE idmascotas = p\_idmascota;

-- Establecer el valor de salida

SET p\_encontrado = (v\_count > 0);

-- Obtener la información

SELECT \*

FROM v\_mascotas\_info

WHERE idmascotas = p\_idmascota;

END

* + - Obtener mascotas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_mascotas`()

BEGIN

SELECT \*

FROM mascotas

ORDER BY idmascotas;

END

* + - Obtener motivos cita

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_motivos\_cita`()

BEGIN

SELECT DISTINCT motivo

FROM citas

ORDER BY motivo;

END

* + - Obtener tratamientos mascotas

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_tratamientos\_mascotas`()

BEGIN

SELECT \*

FROM v\_tratamientos\_mascotas

ORDER BY mascota\_nombre;

END

* + - Obtener tratamientos por mascota

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_tratamientos\_por\_mascota`(

IN p\_idmascota INT,

OUT p\_encontrado BOOLEAN

)

BEGIN

DECLARE v\_count INT;

-- Verificar si existe la mascota

SELECT COUNT(\*) INTO v\_count

FROM mascotas

WHERE idmascotas = p\_idmascota;

-- Establecer el valor de salida

SET p\_encontrado = (v\_count > 0);

IF p\_encontrado THEN

SELECT

t.idtratamientos,

t.diagnostico,

t.fecha\_inicio,

t.fecha\_fin,

m.nombre as nombre\_mascota,

m.idmascotas,

mht.mascotas\_idmascotas,

mht.tratamientos\_idtratamientos,

c.motivo as motivo\_cita,

CONCAT(v.nombre, ' ', v.apellido) as nombre\_veterinario,

v.idveterinarios

FROM tratamientos t

INNER JOIN mascotas\_has\_tratamientos mht ON t.idtratamientos = mht.tratamientos\_idtratamientos

INNER JOIN mascotas m ON mht.mascotas\_idmascotas = m.idmascotas

LEFT JOIN citas c ON t.idtratamientos = c.tratamientos\_idtratamientos

LEFT JOIN veterinarios v ON c.veterinarios\_idveterinarios = v.idveterinarios

WHERE m.idmascotas = p\_idmascota;

END IF;

END

* + - Obtener veterinario especialidad

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_veterinario\_especialidad`(

IN p\_idveterinario INT,

OUT p\_encontrado BOOLEAN

)

BEGIN

DECLARE v\_count INT;

-- Verificar si existe el veterinario

SELECT COUNT(\*) INTO v\_count

FROM veterinarios

WHERE idveterinarios = p\_idveterinario;

-- Establecer el valor de salida

SET p\_encontrado = (v\_count > 0);

IF p\_encontrado THEN

SELECT

vhe.\*,

e.nombre as nombre\_especialidad,

e.descripcion as descripcion\_especialidad,

e.anios\_requeridos

FROM veterinarios\_has\_especialidad vhe

LEFT JOIN especialidad e ON vhe.especialidad\_idespecialidad = e.idespecialidad

WHERE vhe.veterinarios\_idveterinarios = p\_idveterinario;

END IF;

END

* + - Obtener veterinarios info

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_obtener\_veterinarios\_info`()

BEGIN

SELECT

v.idveterinarios,

v.nombre,

v.apellido,

v.telefono,

v.email,

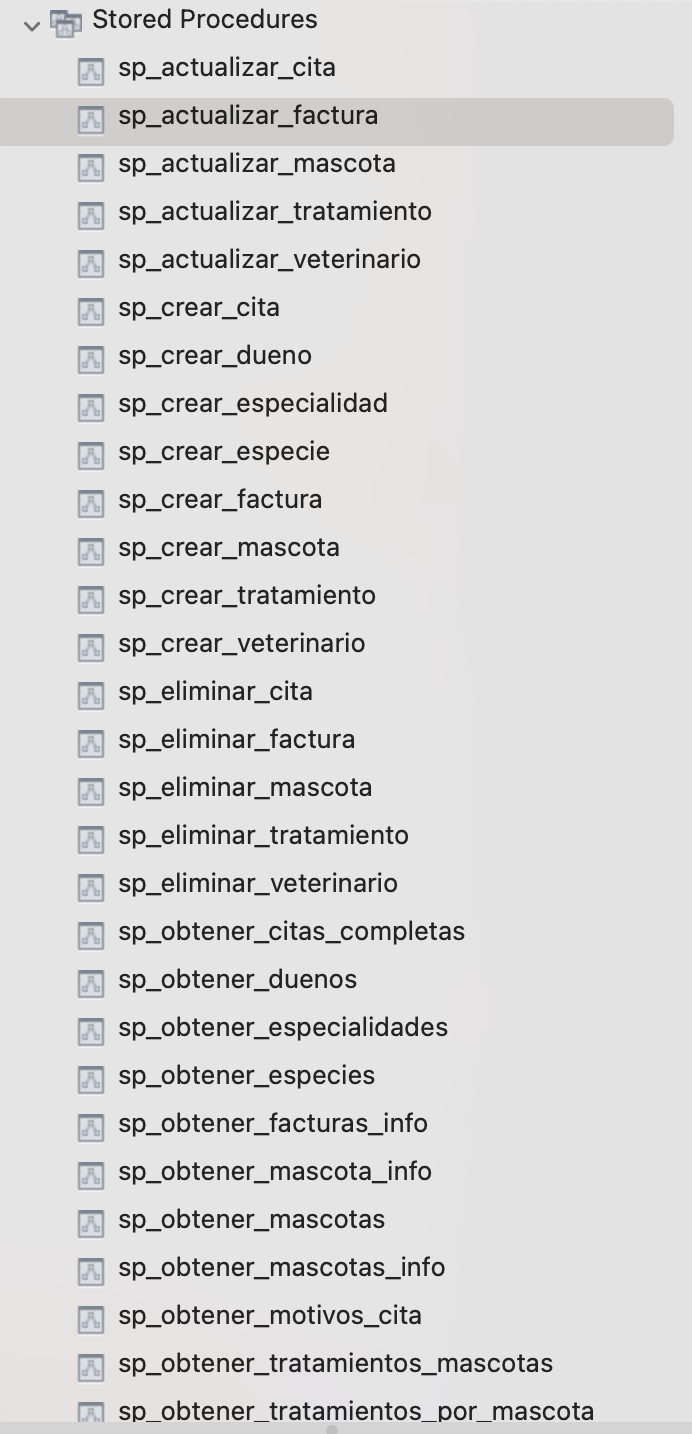
v.especialidades,

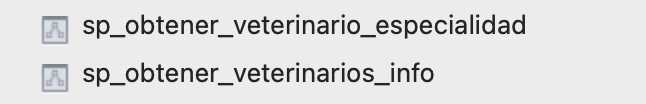
v.especialidad\_ids

FROM v\_veterinarios\_info v

ORDER BY v.nombre, v.apellido;

END





* Front-End

Se utilizó React.js application

* Conclusiones

Durante el desarrollo e implementación de este sistema de base de datos para la clínica veterinaria, se encontraron varios desafíos y aprendizajes significativos:

1. Desafíos de Diseño:
   1. La normalización de las tablas requirió múltiples iteraciones para equilibrar la eficiencia con la usabilidad
   2. El manejo de relaciones muchos a muchos entre mascotas, tratamientos y medicamentos necesitó tablas intermedias cuidadosamente diseñadas
   3. La implementación de claves foráneas y restricciones de integridad referencial fue crucial para mantener la consistencia de los datos
2. Optimización de Consultas:
   1. Se crearon índices estratégicos para mejorar el rendimiento de las consultas más frecuentes
   2. Las vistas materializadas ayudaron a optimizar consultas complejas y frecuentes
   3. Los procedimientos almacenados facilitaron la ejecución de operaciones complejas manteniendo la integridad de los datos
3. Gestión de Transacciones:
   1. Se implementaron transacciones para operaciones críticas como la creación de citas y facturas
   2. Los triggers ayudaron a mantener la consistencia en operaciones como actualización de inventario de medicamentos
   3. Se establecieron puntos de guardado (savepoints) para operaciones complejas
4. Aprendizajes Clave:
   1. La importancia de un diseño inicial sólido y bien pensado
   2. La necesidad de balancear la normalización con el rendimiento práctico
   3. El valor de las restricciones y validaciones a nivel de base de datos
   4. La utilidad de las vistas para simplificar consultas complejas
   5. La importancia de la documentación detallada del esquema y las relaciones
5. Mejoras Futuras:
   1. Implementación de particionamiento de tablas para mejor rendimiento con grandes volúmenes de datos
   2. Optimización adicional de índices basada en patrones de uso
   3. Desarrollo de más procedimientos almacenados para operaciones comunes
   4. Implementación de un sistema de archivado para datos históricos
   5. Mejora en los mecanismos de auditoría y seguimiento de cambios
6. Consideraciones de Seguridad:
   1. Se implementaron roles y permisos específicos para diferentes tipos de usuarios
   2. Se establecieron políticas de respaldo y recuperación
   3. Se implementó logging para operaciones críticas
   4. Se establecieron procedimientos para el manejo de datos sensibles

Esta experiencia ha demostrado la importancia de un diseño de base de datos robusto y bien planificado en el éxito de un sistema de gestión. Las decisiones tomadas en la estructura de la base de datos impactaron significativamente en la funcionalidad, rendimiento y mantenibilidad del sistema completo.