CODERHOUSE

Curso SQL

Clase 10 - Workshop I

Repaso y ejemplo de lo aprendido hasta ahora

Profesor: César Aracena

Índice

Definición del Proyecto

¿Qué modelo de negocio utilizaré?

El modelo para utilizar será el de una aplicación para dispositivos que brindará la posibilidad de adoptar mascotas, información acerca de refugios que den animales en adopción e información sobre veterinarias, qué tipos de servicios brindan y productos y medicamentos.

¿Cuál es el objetivo?

Brindar ayuda para la adopción de mascotas e información para su cuidado

¿Cuál es la necesidad por cubrir?

Llevar un buen registro de los usuarios, adopciones y animales que se registren en la app.

Diagrama Entidad-Relación (conceptual)

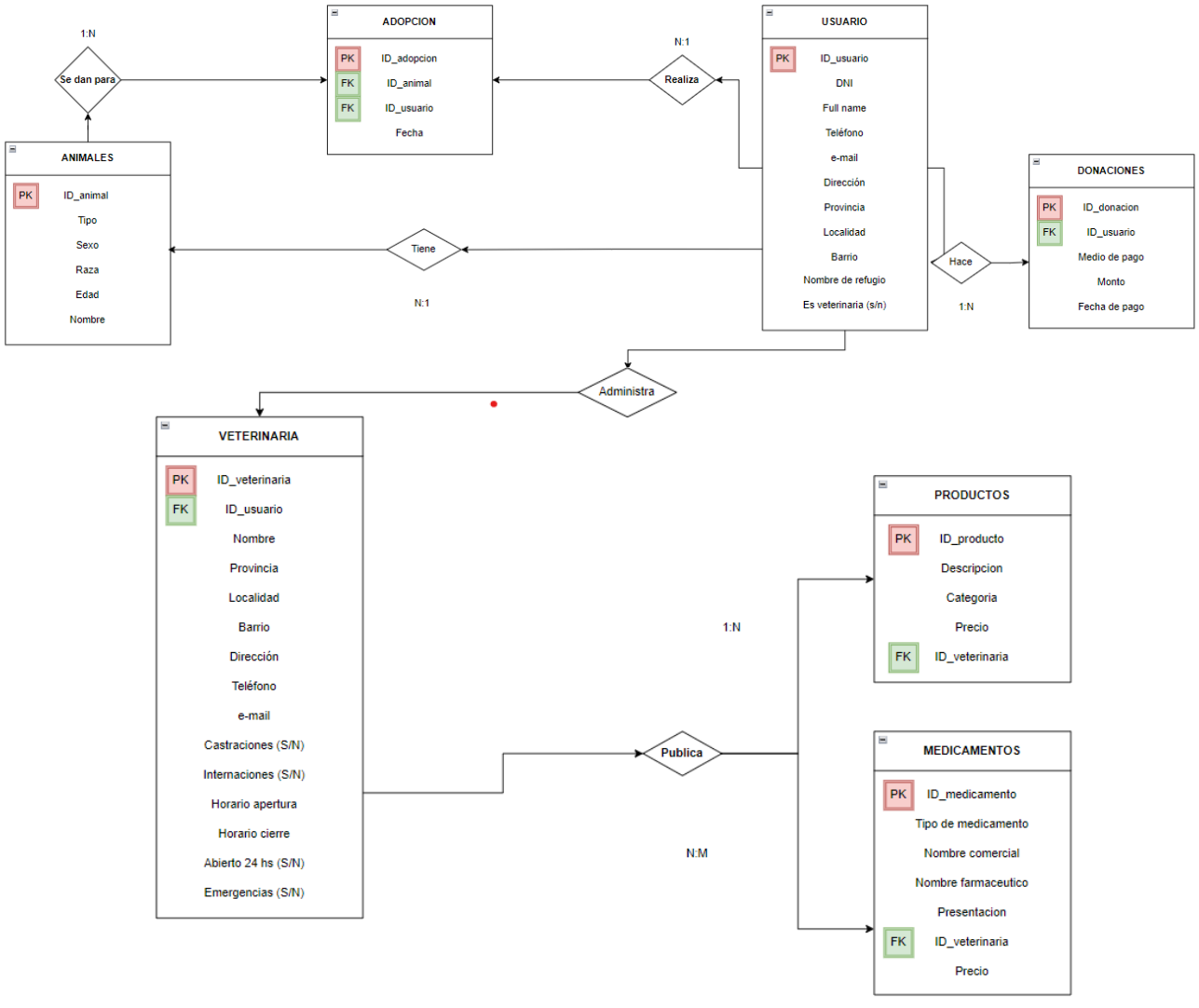
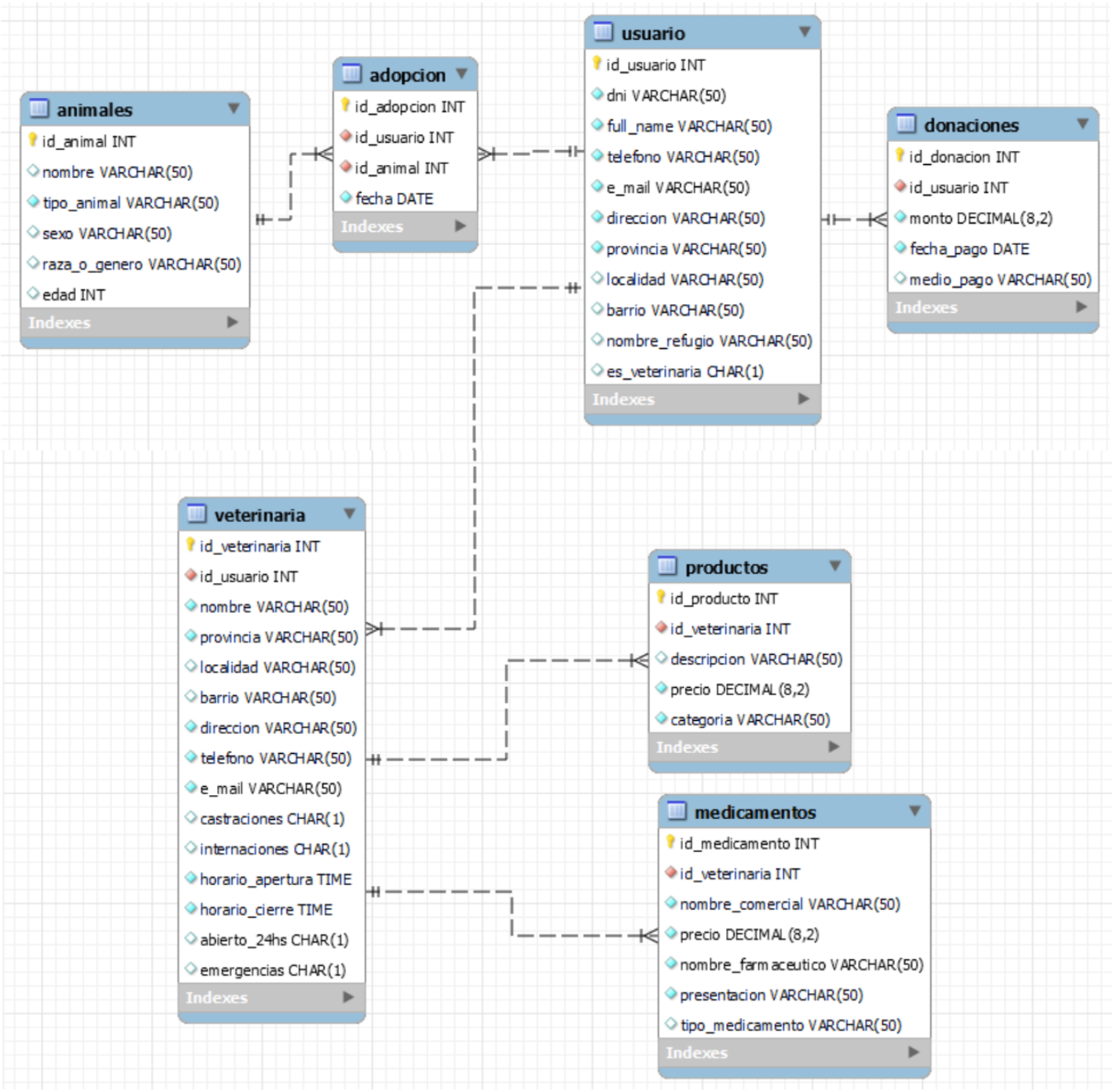


Diagrama Entidad-Relación (Esquemático)



Tablas y sus detalles

TABLA	CAMPO ABREVIADO	NOMBE DEL CAMPO COMPLETO	CLAVE RPIMARIA (PK)	CLAVE FORÁNEA (FK)	TIPO DE DATO
USUARIO	ID_usuario	Identificador del usuario	PK		INT
	dni	Documento de identidad del usuario			VARCHAR (50)
	full_name	Nombre completo del usuario			VARCHAR (50)
	telefono	Teléfono del usuario			VARCHAR (50)
	e-mail	E-mail del usuario			VARCHAR (50)
	direccion	Dirección del usuario			VARCHAR (50)
	provincia	Provincia de residencia del usuario			VARCHAR (50)
	localidad	Localidad de residencia del usuario			VARCHAR (50)
	barrio	Barrio de residencia del usuario			VARCHAR (50)
	nombre_refugio	Nombre del refugio (en caso de serlo)			VARCHAR (50)
	es_veterinaria	Es veterinaria (S/N)			CHAR (1)
ANIMALES	ID_animal	Identificador del animal	PK		INT
	nombre	Nombre del animal			VARCHAR (50)
	tipo_animal	Tipo de animal			VARCHAR (50)
	sexo	Sexo del animal			VARCHAR (50)
	raza	Raza del animal			VARCHAR (50)
	edad	Edad del animal			INT
ADOPCIÓN	ID_adopcion	Identificador de adopción	PK		INT
	ID_usuario	Identificador del usuario		FK	INT
	ID_animal	Identificador del animal		FK	INT
	fecha	Fecha de adopción			DATE
DONACIONES	ID_donacion	Identificador de donación	PK		INT
	ID_usuario	Identificador del usuario		FK	INT
	monto	Monto total donado			DECIMAL (8,2)
	fecha_pago	Fecha de ingreso de pago			DATE
	medio_pago	Medio de pago utilizado			VARCHAR (50)
VETERINARIA	ID_veterinaria	Identificador de veterinaria	PK		INT
	ID_usuario	Identificador de usuario		FK	INT
	nombre	Nombre de la veterinaria			VARCHAR (50)
	provincia	Provincia de residencia de la veterinaria			VARCHAR (50)
	localidad	Localidad de residencia de la veterinaria			VARCHAR (50)
	barrio	Barrio de residencia de la veterinaria			VARCHAR (50)
	direccion	Dirección de residencia de la veterinaria			VARCHAR (50)
	telefono	Teléfono de la veterinaria			VARCHAR (50)
	e-mail	E-mail de la veterinaria			VARCHAR (50)
	castraciones	Realiza castraciones (S/N)			CHAR (1)
	internaciones	Realiza internaciones (S/N)			CHAR (1)
	horario_apertura	Horario de apertura			TIME
	horario_cierre	Horario de cierre			TIME
	abierto_24hs	Abre las 24hs (S/N)			CHAR (1)
	emergencias	Atiende emergencias sin turno previo (S/N)			CHAR (1)
PRODUCTOS	ID_producto	Identificador del producto	PK		INT
	ID_veterinaria	Identificador de veterinaria		FK	INT
	descripcion	Descripción del producto			VARCHAR (50)
	precio	Precio del producto			DECIMAL (8,2)
	categoria	Categoría del producto			VARCHAR (50)
MEDICAMENTOS	ID_medicamento	Identificador del medicamento	PK		INT
	ID_veterinaria	Identificador de veterinaria		FK	INT
	nombre_comercial	Nombre comercial del medicamento			VARCHAR (50)
	nombre_farmaceutico	Nombre farmacéutico del medicamento			VARCHAR (50)
	presentacion	Presentación del medicamento			VARCHAR (50)
	tipo_medicamento	Tipo de medicamento			VARCHAR (50)
	precio	Precio del medicamento			DECIMAL (8,2)

Creación de tablas

```
DROP DATABASE IF EXISTS appMascotas;
CREATE DATABASE appMascotas;
USE appmascotas;
```

Usuario

```
CREATE TABLE IF NOT EXISTS usuario (
id_usuario INT NOT NULL AUTO_INCREMENT,
dni VARCHAR(50) NOT NULL,
full_name VARCHAR(50) NOT NULL,
telefono VARCHAR(50) NOT NULL,
e_mail VARCHAR(50) NOT NULL,
direccion VARCHAR(50) NOT NULL,
provincia VARCHAR(50) NOT NULL,
localidad VARCHAR(50),
barrio VARCHAR(50),
nombre_refugio VARCHAR(50),
es_veterinaria CHAR(1),
PRIMARY KEY (id_usuario)
);
```

Animales

```
CREATE TABLE IF NOT EXISTS animales (
id_animal INT NOT NULL AUTO_INCREMENT,
nombre VARCHAR(50),
tipo_animal VARCHAR(50) NOT NULL,
sexo VARCHAR(50),
raza_o_genero VARCHAR(50),
edad INT,
PRIMARY KEY (id_animal)
);
```

Adopcion

```
CREATE TABLE IF NOT EXISTS adopcion (
id_adopcion INT NOT NULL AUTO_INCREMENT,
id_usuario INT NOT NULL,
id_animal INT NOT NULL,
fecha DATE NOT NULL,
PRIMARY KEY (id_adopcion),
```

```
FOREIGN KEY (id_usuario) REFERENCES usuario (id_usuario) ON DELETE RESTRICT ON
UPDATE CASCADE,
FOREIGN KEY (id_animal) REFERENCES animales (id_animal) ON DELETE RESTRICT ON
UPDATE CASCADE
);
```

Donaciones

```
CREATE TABLE IF NOT EXISTS donaciones (
id_donacion INT NOT NULL AUTO_INCREMENT,
id_usuario INT NOT NULL,
monto DECIMAL(8,2) NOT NULL,
fecha_pago DATE NOT NULL,
medio_pago VARCHAR(50),
PRIMARY KEY (id_donacion),
FOREIGN KEY (id_usuario) REFERENCES usuario (id_usuario) ON DELETE RESTRICT ON
UPDATE CASCADE
);
```

Veterinaria

```
CREATE TABLE IF NOT EXISTS veterinaria (
id_veterinaria INT NOT NULL AUTO_INCREMENT,
id_usuario INT NOT NULL,
nombre VARCHAR(50) NOT NULL,
provincia VARCHAR(50) NOT NULL,
localidad VARCHAR(50),
barrio VARCHAR(50),
direccion VARCHAR(50) NOT NULL,
telefono VARCHAR(50) NOT NULL,
e_mail VARCHAR(50) NOT NULL,
castraciones CHAR(1),
internaciones CHAR(1),
horario_apertura TIME NOT NULL,
horario_cierre TIME NOT NULL,
abierto_24hs CHAR(1),
emergencias CHAR(1),
PRIMARY KEY (id_veterinaria),
FOREIGN KEY (id_usuario) REFERENCES usuario (id_usuario) ON DELETE RESTRICT ON
UPDATE CASCADE
);
```

Productos

```
CREATE TABLE IF NOT EXISTS productos (
id_producto INT NOT NULL AUTO_INCREMENT,
id_veterinaria INT NOT NULL,
descripcion VARCHAR(50),
precio DECIMAL(8,2) NOT NULL,
categoria VARCHAR(50) NOT NULL,
PRIMARY KEY (id_producto),
FOREIGN KEY (id_veterinaria) REFERENCES veterinaria (id_veterinaria) ON DELETE
RESTRICT ON UPDATE CASCADE
);
```

Medicamentos

```
CREATE TABLE IF NOT EXISTS medicamentos (
id_medicamento INT NOT NULL AUTO_INCREMENT,
id_veterinaria INT NOT NULL,
nombre_comercial VARCHAR(50) NOT NULL,
precio DECIMAL(8,2) NOT NULL,
nombre_farmaceutico VARCHAR(50) NOT NULL,
presentacion VARCHAR(50) NOT NULL,
tipo_medicamento VARCHAR(50),
PRIMARY KEY (id_medicamento),
FOREIGN KEY (id_veterinaria) REFERENCES veterinaria (id_veterinaria) ON DELETE
RESTRICT ON UPDATE CASCADE
);
```

Creación de tablas (Paso a paso)

1. Creación de archivos .csv con 10 datos para cada tabla
2. Abrir MySQL Workbench y seleccionar el local host correspondiente
3. En el menú izquierdo en la sección de "Schemas" seleccionar la db "appMascotas"
4. Abir el desplegable de "appMascotas" y luego el de "Tables"
5. Hacer click derecho sobre la tabla "Usuario" y seleccionar la opción "Table Data Import Wizard"
6. Seleccionar la ruta del archivo .csv a importar
7. Seleccionar la opción "Use existing Table" y seleccionar la tabla correspondiente al archivo .csv a importar
8. Corroborar que los nombre de las columnas coincidan y los datos sean correctos. Clickear next>next
9. Comprobar que los datos se hayan cargador correctamente realizando la consulta: select * from appmascotas.usuario;
10. Repetir todos los pasos para cada tabla

Datos de cada tabla

Tabla usuario - GitHub
Tabla animales - GitHub
Tabla adopcion - GitHub
Tabla donaciones - GitHub
Tabla veterinaria - GitHub
Tabla productos - GitHub
Tabla medicamentos - GitHub

Vistas

#1 – v_adopciones

```sql
/*Creo una vista uniendo datos de las tablas adopciones, animales y usuario para poder visualizar
mejor cada adopcion*/

CREATE VIEW v_adopciones AS
SELECT
    adopcion.id_adopcion,
    adopcion.fecha,
    usuario.full_name AS nombre_usuario,
    usuario.telefono AS telefono_usuario,
    animales.nombre AS nombre_animal,
    animales.tipo_animal,
    animales.sexo,
    animales.raza_o_genero,
    animales.edad
FROM adopcion
    JOIN usuario ON adopcion.id_usuario = usuario.id_usuario
    JOIN animales ON adopcion.id_animal = animales.id_animal;

    SELECT * FROM v_adopciones;
```

#2 – v_donaciones

```sql
/*Creo una vista uniendo datos de las tablas usuario y donaciones para visualizar cada
donacion en concreto con los datos mas importantes del usuario*/

CREATE VIEW v_donaciones AS
SELECT
    d.id_donacion,
    d.medio_pago,
    d.monto,
    d.fecha_pago,
    u.full_name AS nombre_usuario,
    u.dni AS dni_usuario,
    u.telefono AS telefono_usuario,
    u.nombre_refugio
FROM donaciones d
JOIN usuario u ON d.id_usuario = u.id_usuario;

 SELECT * FROM v_donaciones;
```

#3 – v_medicamentos

```sql
/*Creo una vista uniendo datos de las tablas medicamentos y veterinaria para mostrar
los fármacos que posee cada veterinaria y sus respectivos datos*/

CREATE VIEW v_medicamentos AS
SELECT
    medicamentos.id_medicamento,
    medicamentos.nombre_comercial,
    medicamentos.nombre_farmaceutico,
    medicamentos.presentacion,
    medicamentos.precio,
    veterinaria.nombre AS nombre_veterinaria,
    veterinaria.telefono AS telefono_veterinaria,
    veterinaria.e_mail AS e_mail_veterinaria
FROM medicamentos
JOIN veterinaria ON medicamentos.id_veterinaria = veterinaria.id_veterinaria;

 SELECT * FROM v_medicamentos;
```

#4 – v_productos

```sql
/*Creo una vista uniendo las tablas productos y veterinaria para mostrar los productos e info
de cada uno y qué veterinarias los tienen disponibles*/

 CREATE VIEW v_productos AS
SELECT
    p.id_producto,
    p.descripcion AS nombre_producto,
    p.categoria,
    p.precio,
    v.nombre AS nombre_veterinaria,
    v.telefono AS telefono_veterinaria,
    v.e_mail AS e_mail_veterinaria
FROM productos p
JOIN veterinaria v ON p.id_veterinaria = v.id_veterinaria;

 SELECT * FROM v_productos;
```

#5 – v_direcciones

```sql
/* Creo una vista para registrar las direcciones de los usuarios que tengan veterinarias registradas
en la app (donde figure la direccion particular del usuario y la direccion donde tiene registrada
su veterinaria)
*/

CREATE VIEW v_direcciones AS
SELECT
    usuario.id_usuario,
    usuario.direccion AS direccion_usuario,
    usuario.provincia AS provincia_usuario,
    usuario.localidad localidad_usuario,
    usuario.barrio AS barrio_usuario,
    veterinaria.id_veterinaria,
    veterinaria.direccion AS direccion_veterinaria,
    veterinaria.provincia AS provincia_veterinaria,
    veterinaria.localidad AS localidad_veterinaria,
    veterinaria.barrio AS barrio_veterinaria
FROM usuario
JOIN veterinaria ON usuario.id_usuario = veterinaria.id_usuario;

   SELECT * FROM v_direcciones;
```

Funciones

#1 – f_encontrar_veterinaria_por_medicamento

```sql
/* Función a la que le indico el nombre de un medicamento y me devuelve
el nombre de la farmacia que lo vende, con el objetivo de facilitar la búsqueda de fármacos específicos
dentro de la DB. Utilizo las tablas veterinaria y medicamentos*/

DELIMITER $$
CREATE FUNCTION `f_encontrar_veterinaria_por_medicamento`(medicamento varchar(50)) RETURNS varchar(50)
    READS SQL DATA
BEGIN
    DECLARE veterinaria VARCHAR (50);
    SET veterinaria = (select
    veterinaria.nombre AS nombre_veterinaria
from veterinaria
join medicamentos ON medicamentos.id_veterinaria = veterinaria.id_veterinaria
where medicamentos.nombre_farmaceutico = medicamento);

RETURN veterinaria;
END $$

select f_encotrar_veterinaria_por_medicamento ('Amoxicilina');
```

#2 – f_contador_animal_sexo

```sql
/* Función que sirve de contador a la que le indico el tipo de animal
y su sexo. Utilizo la tabla animales */

DELIMITER $$
CREATE FUNCTION `f_contador_animal_sexo`(var_tipo_animal VARCHAR(50), var_sexo_animal VARCHAR(50))
RETURNS INT
    READS SQL DATA
BEGIN
    DECLARE cantidad VARCHAR(50);
    SET cantidad =
        (SELECT count(*) from appmascotas.animales
        where sexo = var_sexo_animal
        and tipo_animal = var_tipo_animal);

RETURN cantidad;
END $$

SELECT f_contador_animal_sexo ('Perro', 'Hembra');
```

Stored Procedures

#1 – sp_orden_tabla

```sql
-- Creo un SP para ordenar la tabla seleccionada por un campo determinado y un orden 'ASC' o 'DESC'

DELIMITER $$
CREATE PROCEDURE `sp_orden_tabla` (IN tabla VARCHAR(20), IN campo VARCHAR(50), IN orden VARCHAR(4))
BEGIN
    SET @ordenar = CONCAT('SELECT * FROM',' ', tabla,' ','ORDER BY',' ', campo, ' ', orden);

    PREPARE consulta FROM @ordenar;
    EXECUTE consulta;
    DEALLOCATE PREPARE consulta;

END$$

call  sp_orden_tabla('animales','nombre','desc');
```

#2 – sp_insertar_datos

```sql
-- Creo un SP para la inserción de datos de la tabla Usuario

DROP procedure if exists `sp_insertar_datos`;

DELIMITER $$
CREATE PROCEDURE `sp_insertar_datos`(IN numero_id INT, IN numero_dni VARCHAR(50),
IN nombre VARCHAR(50), IN num_telefono VARCHAR(50), IN email VARCHAR(50),
IN adress VARCHAR(50), IN province VARCHAR(50), IN nom_localidad VARCHAR(50), IN nom_barrio VARCHAR(50),
IN nom_refugio VARCHAR(50), IN es__veterinaria CHAR(1))
BEGIN
    INSERT INTO usuario
    (id_usuario, dni, full_name, telefono, e_mail, direccion, provincia,
    localidad, barrio, nombre_refugio, es_veterinaria)
    VALUES
    (numero_id, numero_dni, nombre, num_telefono, email, adress, province,
    nom_localidad, nom_barrio, nom_refugio, es__veterinaria);
END $$

call sp_insertar_datos(21, '12121212', 'Pedro Louteau', '1158387990', 'usuario@usuario1.com.ar',
'Callao 111', 'Buenos Aires', 'Vicente Lopez', 'Olivos', 'null', 'N');
```

Triggers

Con tabla LOG 'auditoria_animales'
#1 – tr_add_animal y #2 – tr_delete_animal

```sql
-- Creo una tabla LOG donde se almacenarán los datos de los triggers

DROP TABLE IF EXISTS auditoria_animales;

CREATE TABLE auditoria_animales (
id_auditoria_animales INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
id_animal INT NOT NULL,
nombre VARCHAR(50),
tipo_animal VARCHAR(50),
sexo VARCHAR(50),
raza_o_genero VARCHAR(50),
edad INT,
usuario VARCHAR(50),
fecha_hora DATETIME,
accion VARCHAR(50)
);
```

```sql
-- TRIGGER #1
-- Creo un trigger que registre los datos de un nuevo animal ingresado a la DB

DROP trigger if exists `tr_add_animal`;

CREATE TRIGGER `tr_add_animal`
AFTER INSERT ON animales
FOR EACH ROW
INSERT INTO auditoria_animales
VALUES (DEFAULT, new.Id_animal, new.Nombre, new.Tipo_animal, new.Sexo, new.Raza_o_genero, new.Edad, USER(), NOW(),
"Se agrega nuevo animal");

-- Inserto los datos de un nuevo animal en la tabla 'animales'

INSERT INTO animales VALUES (11, 'Toby', 'Perro', 'Macho', 'Gran Danes', 9);

-- Verifico la inserción correcta de datos en ambas tablas

SELECT * from animales;
SELECT * from auditoria_animales;
```

```sql
-- TRIGGER #2
-- Creo un trigger que registre la eliminación de datos de un animal en la tabla 'animales'

DROP trigger if exists `tr_delete_animal`;

CREATE TRIGGER `tr_delete_animal`
BEFORE DELETE ON animales
FOR EACH ROW
INSERT INTO auditoria_animales
VALUES (DEFAULT, OLD.id_animal, OLD.nombre, OLD.tipo_animal, OLD.sexo, OLD.raza_o_genero, OLD.edad, USER(), NOW(),
"Se elimina animal");

-- Elimino datos de la tabla 'animales'

DELETE FROM animales
WHERE id_animal = 11;

-- Verifico la eliminación correcta de datos en ambas tablas

SELECT * from animales;
SELECT * from auditoria_animales;
```

Con tabla LOG 'auditoria_donaciones'
#1 – tr_new_donacion y #2 – tr_delete_donacion

```sql
-- Creo una tabla LOG donde se almacenarán los datos de los triggers

DROP TABLE IF EXISTS auditoria_donaciones;

CREATE TABLE auditoria_donaciones (
id_auditoria_donaciones INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
id_donacion INT NOT NULL,
id_usuario INT NOT NULL,
monto DECIMAL(8,2),
fecha_pago DATETIME,
medio_pago VARCHAR(50),
usuario VARCHAR(50),
fecha_accion DATETIME,
accion VARCHAR(50)
);
```

```sql
-- TRIGGER #1
-- Creo un trigger que registre los datos de una nueva donación

DROP trigger if exists `tr_new_donacion`;

CREATE TRIGGER `tr_new_donacion`
AFTER INSERT ON donaciones
FOR EACH ROW
INSERT INTO auditoria_donaciones
VALUES (DEFAULT, new.Id_donacion, new.Id_usuario, new.monto, new.fecha_pago, new.medio_pago, USER(), NOW(),
"Ingresa nueva donacion");

-- Inserto los datos de una nueva donación en la tabla 'donaciones'

INSERT INTO donaciones VALUES (11, 12, 500.00, '2021-07-23', 'Pay Pal');

-- Verifico la inserción correcta de datos en ambas tablas

SELECT * from donaciones;
SELECT * from auditoria_donaciones;
```

```sql
-- TRIGGER #2
-- Creo un trigger que registre la eliminación de datos de la tabla 'donaciones'

DROP trigger if exists `tr_delete_donacion`;

CREATE TRIGGER `tr_delete_donacion`
BEFORE DELETE ON donaciones
FOR EACH ROW
INSERT INTO auditoria_donaciones
VALUES (DEFAULT, OLD.id_donacion, OLD.id_usuario, OLD.monto, OLD.fecha_pago, OLD.medio_pago, USER(), NOW(), "Se elimina donacion");

-- Elimino datos de la tabla 'donaciones'

DELETE FROM donaciones
WHERE id_donacion = 11;

-- Verifico la eliminación correcta de datos en ambas tablas

SELECT * from donaciones;
SELECT * from auditoria_donaciones;
```

Con tabla LOG ‘auditoria_update_producto
#1 – tr_aumento_precio_producto

```sql
-- Creo una tabla LOG donde se almacenarán los datos del trigger

DROP TABLE IF EXISTS auditoria_update_producto;

CREATE TABLE auditoria_update_producto (
id_auditoria_producto INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
id_producto INT NOT NULL,
descripcion VARCHAR(50),
viejo_precio DECIMAL(8,2),
nuevo_precio DECIMAL(8,2),
usuario VARCHAR(50),
fecha_accion DATETIME,
accion VARCHAR(50)
);
```

```sql
-- TRIGGER
-- Creo un trigger que registre el viejo y nuevo precio de un producto

DROP trigger if exists `tr_aumento_precio_producto`;

CREATE TRIGGER `tr_aumento_precio_producto`
AFTER UPDATE ON productos
FOR EACH ROW
INSERT INTO auditoria_update_producto
VALUES (DEFAULT, OLD.id_producto, OLD.descripcion, OLD.precio, NEW.precio, USER(), NOW(), "Precio actualizado");

-- Modifico el precio de un producto de la tabla 'productos'

UPDATE productos
SET precio = 1000.00
WHERE id_producto = 3;

-- Verifico la modificación correcta del precio en ambas tablas

SELECT * from productos;
SELECT * from auditoria_update_producto;
```