Aplicación “My Pets”

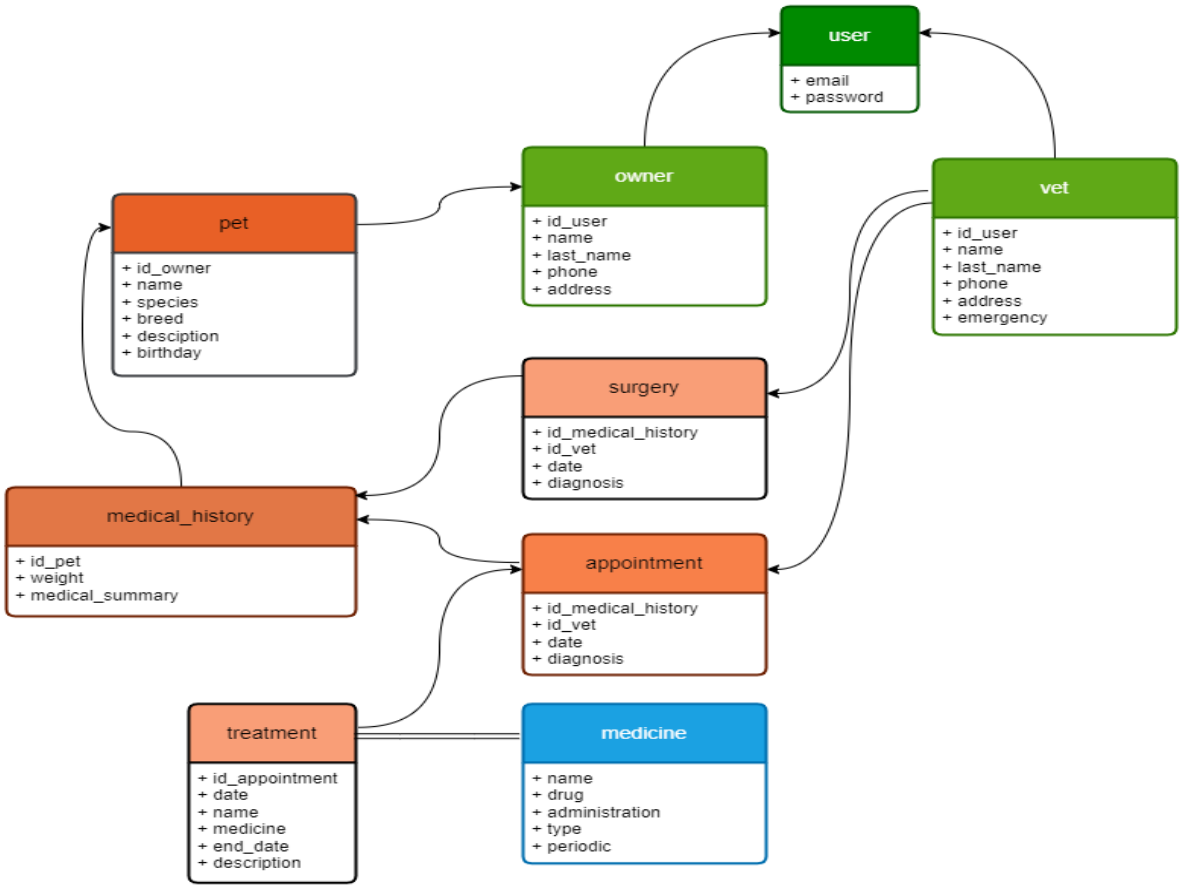
El proyecto a realizar será la base de datos que almacena y organiza la información referente a nuestras mascotas, de manera que en una sola aplicación, podemos guardar todos los detalles de manera fiable, permitiendo agregar, editar y compartir fácilmente los datos más importantes como, vacunación y tratamientos realizados.

Una característica principal será que deberá ser accesible desde cualquier lugar de manera rápida y que tenga la información más relevante, para ser compartida con los veterinarios que atienden a nuestras mascotas.

Uno de los objetivos principales será mantener la información organizada correctamente, buscando que sea fácilmente accesible a los distintos actores de manera controlada y segura, permitiendo un futuro escalamiento de la aplicación que sea base para las futuras funciones, orientadas principalmente a la parte comercial.

La información será suministrada por los mismos usuarios al registrarse como usuarios, donde ingresarán datos personales, de sus mascotas y de equipamiento médico en caso de que corresponda.

La información principal a guardar será respecto a tres principales entidades, mascotas, dueños y veterinarios. Además habrá que almacenar información de segunda importancia, como vacunas, medicamentos, entre otros. Inicialmente se plantea un primer



esquema borrador como el siguiente, el cual espero sea pulido y organizado con el correr de las clases.

Adjunto Link de diagrama:
https://drive.google.com/file/d/19APXJ1ss57vt_fVe2s6M804z0GV5fZIS/view?usp=sharing

Adjunto Link de Tabla:
https://docs.google.com/spreadsheets/d/1gVuutqbPCG0IJtQchQu0hfVm0hKxELPxpGvk4pd9KXE/edit?usp=sharing

Tablas app My Pets						
Tabla	Descripcion	Nombre Campo	Tipo	**PK**	**FK**	Detalle
user	Contiene todos los usuarios registrados sin discriminar roles dentro de la app	**id_user**	**INT**	**X**		**AUTOINCREMENT**
		email	VARCHAR (50)			UNIQUE NOT NULL
		password	VARCHAR (50)			NOT NULL
owner	Tabla de dueños de mascotas, asociada a un usuario en particular	**id_owner**	**INT**	**X**		**AUTOINCREMENT**
		id_user_owner	INT		**X**	NOT NULL
		name	VARCHAR (30)			NOT NULL
		last_name	VARCHAR (30)			NOT NULL
		phone	VARCHAR (15)			NOT NULL
		address	VARCHAR (100)			NOT NULL
vet	Tabla de veterinarios, asociada a un usuario en particular	**id_vet**	**INT**	**X**		**AUTOINCREMENT**
		id_user_vet	INT		**X**	NOT NULL
		name	VARCHAR (30)			NOT NULL
		last_name	VARCHAR (30)			NOT NULL
		phone	VARCHAR (15)			NOT NULL
		address	VARCHAR (100)			NOT NULL
		emergency	INT (BOOLEAN			NOT NULL DEFAULT 0

			)			
pet	Mascotas, asociadas obligatoriamente a un dueño.	**id_pet**	**INT**	**X**		**AUTOINCREMENT**
		id_owner	INT		**X**	NOT NULL
		name	VARCHAR (30)			NOT NULL
		species	VARCHAR (30)			NOT NULL
		description	TEXT			NOT NULL
		birthday	DATE			NOT NULL
		breed	VARCHAR (100)			NOT NULL
medical_history	Ficha de informacion medica de cada mascota, puede relacionarse con atenciones y cirugias de manera opcional	**id_medical_history**	**INT**	**X**		**AUTOINCREMENT**
		id_pet_medical_history	INT		**X**	NOT NULL
		weight	FLOAT			NOT NULL
		medical_summary	VARCHAR (100)			NOT NULL
appointment	Tabla que contiene cada atencion que se le realiza a la mascota, se debe indicar obligatoriamente un veterinario y una historia clinica	**id_appointment**	**INT**	**X**		**AUTOINCREMENT**
		id_vet	INT		**X**	NOT NULL
		id_medical_history	INT		**X**	NOT NULL
		date	DATETIME			NOT NULL
		diagnosis	TEXT			NOT NULL
surgery	Tabla que contiene cada cirugia que se le realiza a la mascota, se debe indicar obligatoriamente un veterinario y una historia clinica	**id_surgery**	**INT**	**X**		**AUTOINCREMENT**
		id_vet	INT		**X**	NOT NULL
		id_medical_history	INT		**X**	NOT NULL
		date	DATETIME			NOT NULL
		diagnosis	TEXT			NOT NULL
treatment	Tabla que contiene cada tratamiento propuesto que se le realiza a la mascota dentro de una atención, por ejemplo, dentro de una atención, se puede indicar una sutura de herida y un inyectable, siendo tratamientos distintos dentro de una	**id_treatment**	**INT**	**X**		**AUTOINCREMENT**
		id_appointment	INT		**X**	NOT NULL
		date	DATETIME			NOT NULL
		name	VARCHAR (50)			NOT NULL
		medicine	VARCHAR (50)			NOT NULL
		end_date	DATETIME			NOT NULL

	misma atencion.	description	TEXT			NOT NULL
treatment_ medicine	Tabla intermedia	**id_treatment_medicine**	**INT**	**X**		**AUTOINCREMENT**
		id_medicine	INT		**X**	NOT NULL
		id_treatment	INT		**X**	NOT NULL
medicine	Tabla con medicamentos y sus características, cargada por admin	**id_medicine**	**INT**	**X**		**AUTOINCREMENT**
		name	VARCHAR (30)			NOT NULL
		drug	VARCHAR (50)			NOT NULL
		administration	VARCHAR (30)			NOT NULL
		type	VARCHAR (30)			NOT NULL
		periodic	INT (BOOLEAN)			NOT NULL DEFAULT 0

Vistas propuestas:

- **dueños_con_mas_mascotas** : View que solo muestra dueños (id, nombre y cantidad de mascotas) que tienen más de una mascota, ordenados descendentemente por cantidad de mascotas. Formada por unión de tablas **pet** y **owner**.
- **dueños_sin_mascotas_para_ofrecer_adopcion** : View que muestra dueños (id, nombre y teléfono) que no tienen mascotas asociadas, pensado para ofrecer obtener datos de contacto para ofrecer adopción. Formada por unión de tablas **pet** y **owner**.
- **mascotas_segun_cantidad_atenciones_nombre_especie** : View que muestra datos de mascota (id, nombre y especie de la mascota) y cantidad de atenciones recibidas. Formada por unión de tablas **pet , medical_history** y **appointment**.
- **drogas_usadas_cantidad_para_reposicion** : View que muestra datos de medicina (id, nombre comercial y droga) y que calcula la cantidad de unidades usadas en las atenciones brindadas a las mascotas. Formada por unión de tablas **medicine** y **treatment_medicine**.
- **cirujanos_que_hacen_emergencias** : View que solo muestra veterinarios (Nombre, apellido y teléfono) que tengan experiencia en cirugía y que además realicen urgencias. Formada por unión de tablas **vet** y **surgery**.

Funciones propuestas:

- **Buscar_cirujano_mas_experimentado_por_apellido**: Función que recibe un VARCHAR(30) por parámetro (representando el apellido o algunos caracteres del mismo), y devuelve un mensaje con el apellido del primer cirujano encontrado. Además indica la cantidad de cirugías realizadas. En caso de no encontrar, retorna el mensaje “No hay un cirujano con un apellido que contenga el parámetro ‘input’ ”. Formada por unión de tablas **surgery** y **vet**.
- **validar_stock_por_id_medicine** : Función que recibe integers que corresponden a id_medicine, Stock inicial y stock actual, luego verifica que el stock actual es correcto teniendo en cuenta lo que había inicialmente menos lo que se gastó en las atenciones. Devuelve 1 si el stock es correcto o 0 si no lo es. Formada por unión de tablas **medicine** y **treatment_medicine**, mediante la vista “**drogas_usadas_cantidad_para_reposicion**”.

Stored procedures:

- **obtener_tabla_ordenada_por**: Procedimiento para consultar una tabla por nombre y ordenarla de forma ascendente o descendente por un determinado campo. Los campos son nombre de tabla, campo, orden. El nombre de tabla es obligatorio, pero el resto son opcionales.
    - call my_pets.obtener_tabla_ordenada_por(‘ ‘, ‘ ‘‘; ‘);
        - Retorna una excepción al código 1644, con el mensaje: “Necesitamos al menos el nombre de la tabla!”
    - call my_pets.obtener_tabla_ordenada_por('medicine', ‘ ‘', ‘ ‘);
    - call my_pets.obtener_tabla_ordenada_por('medicine', 'drug', ‘ ‘);
    - call my_pets.obtener_tabla_ordenada_por('medicine', 'drug', 'desc');
- **eliminar_owners_sin_mascotas** : Procedimiento para eliminar todos aquellos owners que no tengan mascotas asociadas, además elimina los users asociados a esos owners. Está pensado para ser un procedimiento de limpieza de tabla. Utiliza las tablas **owner**, **user** y **pet**. Se debe llamar sin parametros:
    - call my_pets.eliminar_owners_sin_mascotas();

Triggers:

- **log_appointment_insert** : Trigger que se dispara en los inserts en tabla appointment, y luego de agregar el registro, crea un nuevo registro en tabla log_appointment con datos de operación (usuario que realiza la acción, fecha y horario de inserción) y con datos del nuevo registro (id_appointment , id_vet , id_pet, app_date y diagnosis). La idea es mantener una copia de los registros de la tabla.
- **log_appointment_update** : Trigger que se dispara en los updates en tabla appointment, y antes de modificar el registro, crea un nuevo registro en tabla

log_appointment con datos de operación (usuario que realiza la acción, fecha y horario de inserción) y con datos del nuevo registro (id_appointment , id_vet , id_pet, app_date y diagnosis). La idea es mantener un registro de los cambios realizados en la tabla.

- **log_medicine_insert** : Trigger que se dispara en los inserts en tabla medicine, y despues de insertar el registro, crea un nuevo registro en tabla log_medicine con datos de operación (usuario que realiza la accion, fecha y horario de inserción) y con datos del nuevo registro (id_medicine , name, drug, administration, type y periodic). La idea es mantener un registro de los cambios realizados en la tabla.
- **log_medicine_update** : Trigger que se dispara en las modificaciones que se realicen en tabla medicine, y antes de modificar el registro, crea un nuevo registro en tabla log_medicine con datos de operación (usuario que realiza la acción, fecha y horario de inserción) y con todos los datos del registro original (id_medicine , name, drug, administration, type y periodic).La función es mantener un registro y copia de los datos y su modificación
- **log_medicine_delete** : Trigger que se dispara en las eliminaciones que se realicen en tabla medicine, y antes de realizarlo, crea un nuevo registro en tabla log_medicine con datos de operación (usuario que realiza la acción, fecha y horario de inserción) y con todos los datos del registro original (id_medicine , name, drug, administration, type y periodic). La idea es mantener una copia de los registros de la tabla, en caso de ser necesario.
-