**JAVA - PLANEACIÓN COMPLETA**

**(Proyecto java)**

**ALEJANDRA MACHUCA MOLINA**

**YESSICA ANDREA PEREZ MACHUCA**

**T2**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS**

**SANDBOX**

**TIBU**

**2024**

Tabla de Contenidos

Introducción 1

Caso de Estudio 2

Planificación 3

Construcción del Modelo Conceptual 4

Descripción 5

Descripción técnica 6

Gráfica 7

Construcción del Modelo Lógico 8

Descripción 9

Gráfica 10

Construcción del Modelo Físico 11

Descripción 12

Gráfica 13

Tablas 14

Relaciones entre Tablas 15

Inserción de Datos 16

Controladores 17

Modelos 18

Vistas 19

Clases 20

**Introducción**

En la actualidad, la gestión de datos es esencial para cualquier empresa que trabaje con varios servicios, en este caso Happy Feet ya que nos permite el manejo adecuado de los datos, los cuales son: Pets, Owners, Additional\_Services, Doctors, Consult, Procedures, Vaccine\_History, Health\_History, Appointments, Medicines, Products, Inventary, Administrator, Details, Invoice, Adoption, Visit\_History, este proyecto tiene como objetivo la construcción de un sistema de gestión de una veterinaria.

El sistema propuesto permitirá a la empresa almacenar información clave de la empresa sobre las entidades. Además se garantiza la integridad y consistencia de los datos.

**Caso de Estudio**

Considerando que Happy Feet desea implementar un sistema para el manejo adecuado de las operaciones de la clínica veterinaria Veterinaria\_Patitas\_Felices. La empresa necesita una solución que le permita registrar, actualizar y consultar información sobre sus entidades.

1. Pets: Registro de datos de una mascota con sus demás atributos.
2. Owners: Registro de datos de un dueño con sus mascotas y demás atributos.
3. Addtitonal\_Services: Servicios adicionales que desea un dueño para su mascota.
4. Doctors: Registro de datos de un doctor con sus demás atributos.
5. Consult: Datos de una consulta para mascotas y demás atributos de la entidad consult.
6. Procedures: Tipos de procedimientos realizados en la veterinaria.
7. Vaccine\_History: Historial de vacunas para una mascota.
8. Health\_History: Historial de salud y condiciones para una mascota.
9. Appointments: Citas agendadas para las mascotas.
10. Medicines: Medicamentos disponibles en la veterinaria.
11. Products: Productos disponibles en la veterinaria.
12. Inventary: Inventario de la veterinaria.
13. Administrator: Será el encargado de generar varias consultas y procedimientos.
14. Details: Detalles de las facturas de la empresa.
15. Invoice: Facturas generadas por la empresa.
16. Adoption: Adopción de mascotas.

El sistema debe facilitar las citas que pida un dueño, y sus diferentes servicios adicionales.

**Planificación**

**Construcción del modelo conceptual**

El modelo conceptual es el primer paso en el diseño de una base de datos, donde se identifican las entidades principales, atributos y las relaciones entre ellas.

**Descripción**

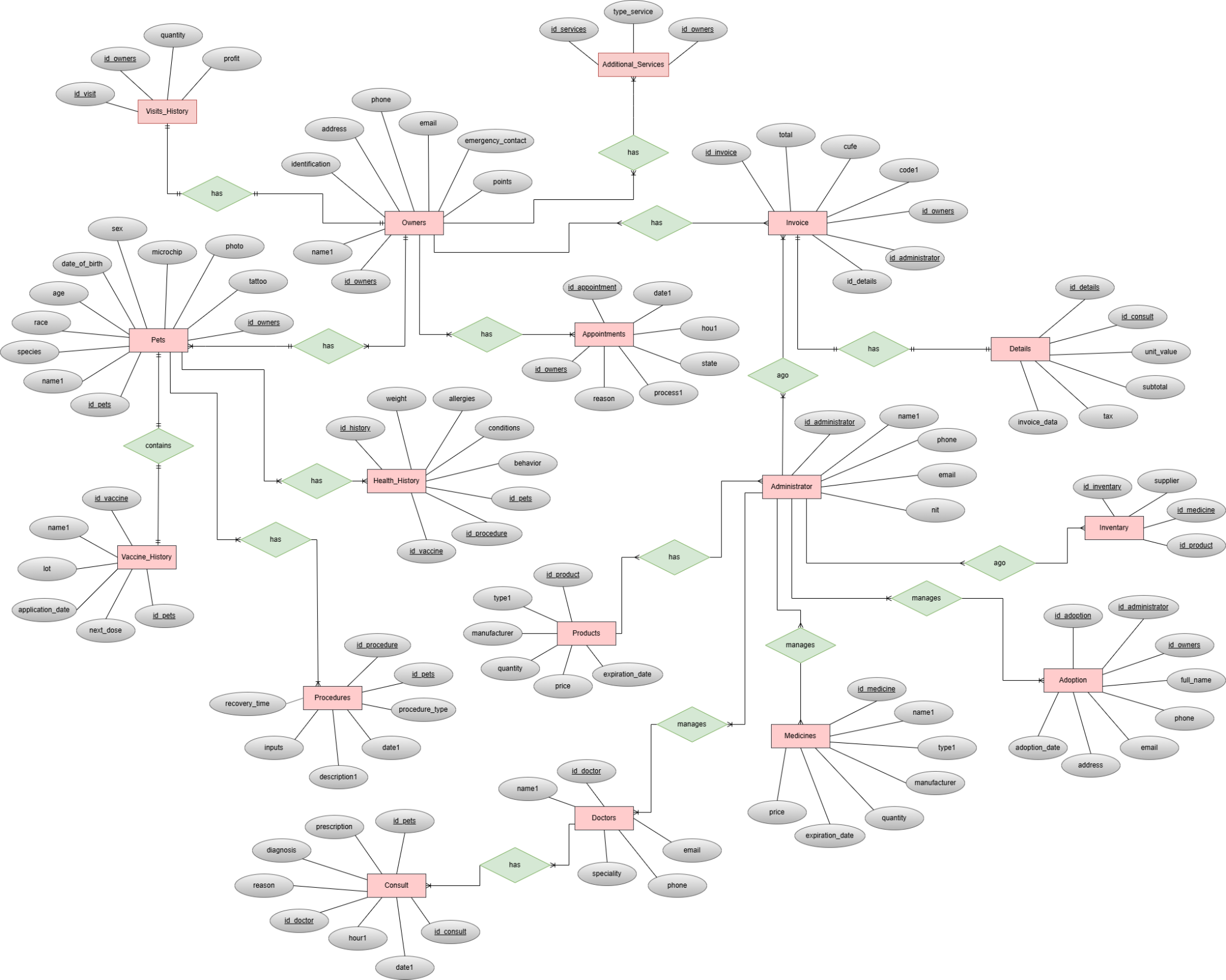
Se realizó el modelo conceptual identificando cada una de las entidades, sus atributos y las relaciones entre ellas, este modelo nos permite obtener una visión clara y estructurada.

**Descripción Técnico**

1. **Entidades - Atributos**

* **Owners**
* **id\_owners:** Identificador único del dueño.
* **name1:** Nombres completos del dueño.
* **identification:** Identificación del dueño.
* **address:** Dirección del dueño.
* **phone:** Número celular del dueño.
* **email:** Correo electrónico del dueño.
* **emergency\_contact:** Contacto de emergencia del dueño.
* **points:** Puntos obtenidos por el dueño.
* **password1:** Contraseña del dueño.
* **Pets**
* **id\_pets:** Identificador único de la mascota.
* **name1:** Nombre de la mascota.
* **species:** Especie de la mascota.
* **race:** La raza de la mascota.
* **age:** Edad de la mascota.
* **date\_of\_birth:** Fecha de nacimiento de la mascota.
* **sex:** Sexo de la mascota.
* **microchip:** Si tiene o no microchip.
* **photo:** Foto de la mascota.
* **tattoo:** Si tiene o no un tattoo.
* **id\_owners:** Identificador único del dueño (FK).
* **Vaccine\_History**
* **id\_vaccine:** Identificador único del historial de vacunas.
* **name1:** Nombre del historial de vacunas.
* **lot:** Lote de la vacuna.
* **application\_date:** Fecha de aplicación de la vacuna.
* **next\_dose:** Fecha de aplicación de la próxima dosis de la vacuna.
* **id\_pets:** Identificador único de la mascota.
* **Consult**
* **id\_consult:** Identificador único de la consulta.
* **date1:** Fecha de la consulta.
* **hour1:** Hora de la consulta.
* **reason:** Motivo de la consulta.
* **diagnosis:** Diagnóstico de la consulta.
* **prescription:** Prescripción enviada por el doctor.
* **id\_doctor:** Identificador único del doctor.
* **id\_pets:** Identificador único de la mascota.
* **Doctors**
* **id\_doctor:** Identificador único del doctor.
* **name1:** Nombre del doctor.
* **speciality:** Especialidad del doctor.
* **phone:** Número de celular del doctor.
* **email:** E-mail del doctor.
* **password1:** Contraseña del doctor.
* **Medicines**
* **id\_medicines:** Identificador único de medicinas.
* **name1:** Nombre de las medicinas.
* **type1:** Tipo de medicamento.
* **manufacturer:** Fabricante del medicamento.
* **quantity:** Cantidad de medicamentos.
* **expiration\_date:** Fecha de vencimiento del medicamento.
* **price:** Precio del medicamento.
* **Procedures**
* **id\_procedure:** Identificador único de procesos.
* **procedure\_type:** Tipo de procedimientos.
* **date1:** Fecha del procedimiento.
* **description1:** Descripción del procedimiento.
* **inputs:** Entradas a un procedimiento.
* **recovery\_time:** Tiempo de recuperación del procedimiento.
* **id\_pets:** Identificador único de la mascota.
* **Inventary**
* **id\_inventory:** Identificador único del inventario.
* **supplier:** Proveedor de los medicamentos.
* **id\_medicine:** Identificador único de medicamentos (FK).
* **id\_product:** Identificador único de productos (FK).
* **Appointment**
* **id\_appointment:** Identificador único citas.
* **date1:** Fecha de la cita.
* **hour1:** Hora de la cita.
* **state:** Estado de la cita.
* **process1:** Proceso realizado en la cita.
* **reason:** Motivo de la consulta.
* **id\_owners:** Identificador único de dueños (FK).
* **Administrator**
* **id\_administrator:** Identificador único de administradores.
* **name1:** Nombre del administrador.
* **phone:** Número celular del administrador.
* **email:** E-mail del administrador.
* **nit:** Nit de la empresa.
* **password1:** Contraseña del administrador.
* **Health\_History**
* **id\_vaccine:** Identificador único del historial de salud.
* **name1:** Nombre del historial de salud.
* **lot:** Lote del historial de salud.
* **application\_date:** Fecha de aplicación al historial de salud.
* **next\_dose:** Proxima dosis de medicamentos en el historial de salud.
* **id\_pets:** Identificador único de las mascotas (FK).
* **Visit\_History**
* **id\_visit:** Identificador único de visitas.
* **quantity:** Cantidad de visitas a la veterinaria.
* **profit:** Beneficios por asistir a la visita (en cierta cantidad de visitas).
* **id\_owners:** Identificador único de los dueños.
* **Invoice**
* **id\_invoice:** Identificador único de la factura.
* **total:** Total de los productos comprados.
* **cufe:** Identificador de la factura (códigos).
* **code1:** Código qr de la factura.
* **id\_owners:** Identificador único del dueño (FK).
* **id\_administrator:** Identificador único del administrador (FK).
* **id\_details:** Identificador único de los detalles (FK).
* **Details**
* **id\_details:** Identificador único de detalles.
* **unit\_value:** Valor por unidad de los productos.
* **subtotal:** Subtotal de cada producto.
* **tax:** Impuesto por producto.
* **invoice\_date:** Fecha de la factura.
* **id\_consult:** Identificador único de la consulta (FK).
* **Adoption**
* **id\_adoption:** Identificador único de adopción.
* **full\_name:** Nombre del dueño que adoptó.
* **phone:** Número celular del dueño que adoptó.
* **email:** E-mail del dueño que adoptó.
* **address:** Dirección de dueño que adoptó.
* **adoption\_date:** Fecha de la adopción.
* **id\_administrator:** Identificador único de administradores (FK).
* **id\_owners:** Identificador único de los dueños (FK).
* **Products**
* **id\_products:** Identificador único de productos.
* **type1:** Tipo de productos.
* **manufacture:** Proveedor de los productos.
* **quantity:** Cantidad de los productos.
* **expiration\_date:** Fecha de vencimiento del producto.
* **price:** Precio por producto.
* **Additional\_Services**
* **id\_services:** Identificador único de los servicios adicionales.
* **type:service:** Tipo del servicio adicional.
* **id\_owners:** Identificador único de los dueños (FK).

1. **Gráfica**

****

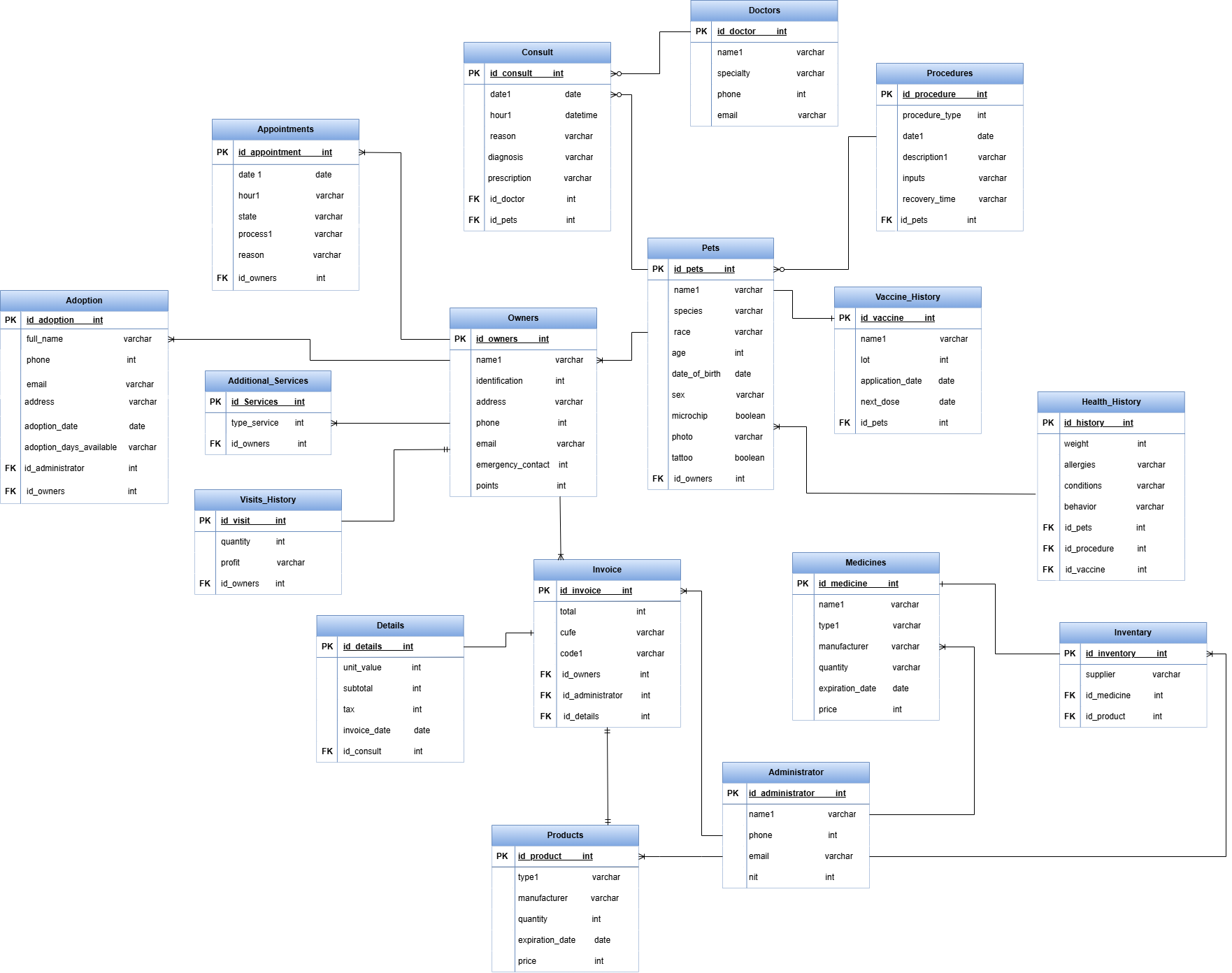
**Construcción del modelo lógico**

El modelo lógico de una base de datos transforma el modelo conceptual en una estructura más detallada, donde se definen las claves primarias, claves foráneas, y las relaciones entre las tablas. En esta fase, se comienza a definir cómo se almacenarán los datos en la base de datos relacional, preservando la integridad y las relaciones establecidas en el modelo conceptual.

**Descripción**

El **Sistema** será modelado lógicamente con las siguientes tablas: Pets, Owners, Doctors, Administrator, Invoice, Inventory, Additional\_Services, Consult, Procedures, Vaccine\_History, Health\_history, Appointments, Medicines, Products, Details, Adoption, Visits\_History. A través de estas tablas se mantendrán las relaciones y se definirán las claves necesarias para la integridad referencial.

**Gráfica**

****

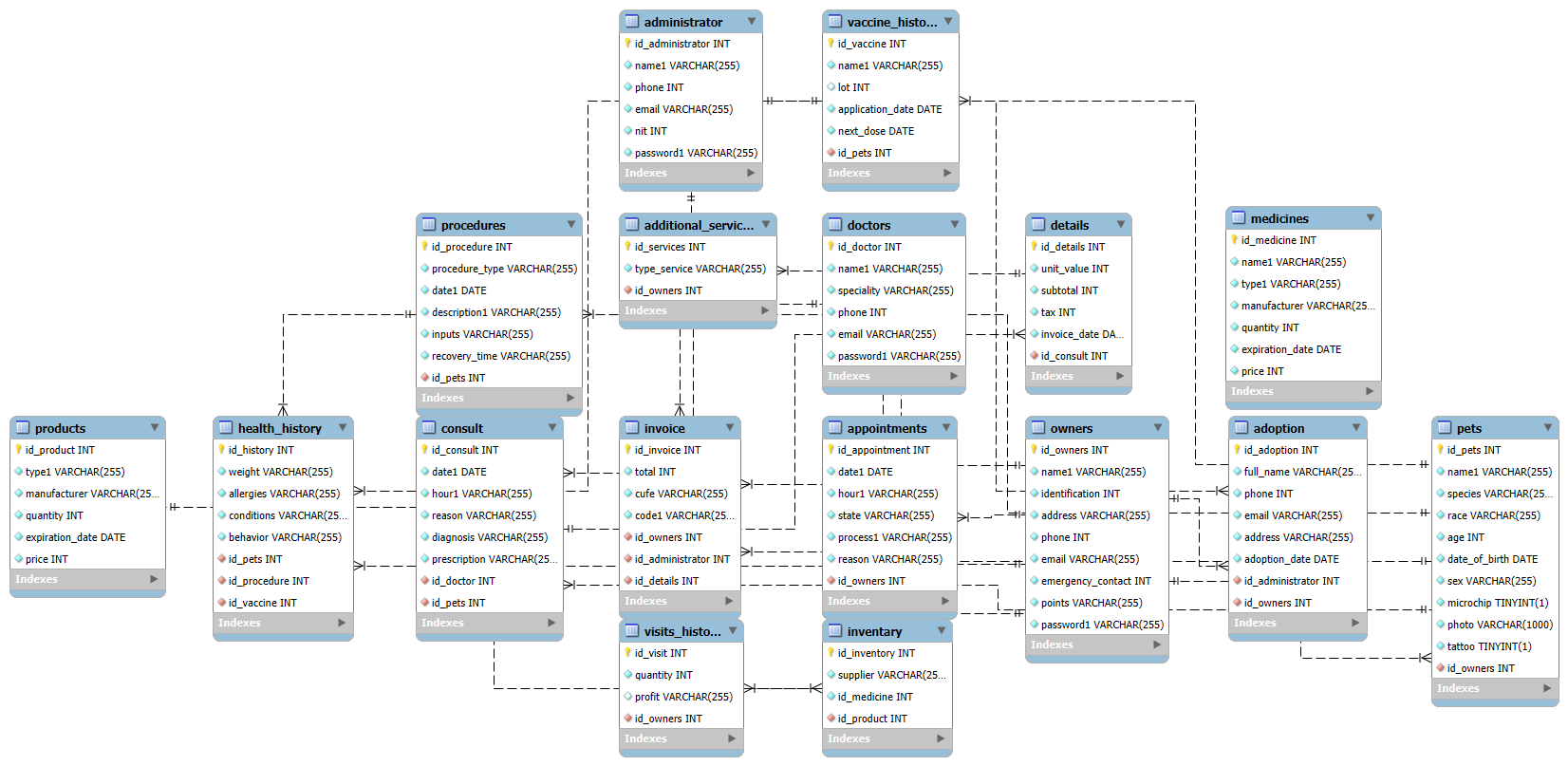
**Construcción del modelo físico**

Se diseñó el modelo físico considerando el modelo lógico que incluye todas las entidades, sus atributos y las relaciones entre ellas. Además, este modelo incorpora los tipos de datos de los atributos previamente definidos.

**Descripción**

El modelo físico se diseñó para funcionar en MySQL, donde cada entidad se representa como una tabla compuesta por sus atributos correspondientes, organizados en columnas con tipos de datos específicos según sea necesario.

**Gráfica**

****

**Tablas**

* Para crear la base de datos utilice el siguiente comando:

create database Veterinaria\_Patitas\_Felices;

* Para utilizar la base de datos ocupe el siguiente comando:

use Veterinaria\_Patitas\_Felices;

* Comenzaremos creando las tablas junto con sus tipos de datos correspondientes. Para esto, utiliza los siguientes comandos:

1. Creación de la tabla Owners:

create table Owners(

id\_owners int not null auto\_increment primary key,

name1 varchar(255) not null,

identification int(15) not null,

address varchar(255) not null,

phone int(15) not null,

email varchar(255) not null,

emergency\_contact int(15) not null,

points varchar(255) not null,

password1 varchar(255) not null

);

1. Creación de la tabla Additional\_Sevices:

create table Additional\_Services(

id\_services int(15) not null auto\_increment primary key,

type\_service varchar(255) not null,

id\_owners int(15) not null,

foreign key(id\_owners) references Owners(id\_owners) on delete cascade

);

1. Creación de la tabla Pets:

create table Pets(

id\_pets int(15) not null auto\_increment primary key,

name1 varchar(255) not null,

species varchar(255) not null,

race varchar(255) not null,

age int(15) not null,

date\_of\_birth date not null,

sex varchar(255) not null,

microchip boolean not null,

photo varchar(1000) not null,

tattoo boolean not null,

id\_owners int(15) not null,

foreign key(id\_owners) references Owners (id\_owners) on delete cascade

);

1. Creación de la tabla Doctors:

create table Doctors(

id\_doctor int(15) not null auto\_increment primary key,

name1 varchar(255) not null,

speciality varchar(255) not null,

phone int(15) not null,

email varchar(255) not null,

password1 varchar(255) not null

);

1. Creación de la tabla Consult:

create table Consult(

id\_consult int(15) not null auto\_increment primary key,

date1 date not null,

hour1 varchar(255) not null,

reason varchar(255) not null,

diagnosis varchar(255) not null,

prescription varchar(255) not null,

id\_doctor int(15) not null,

id\_pets int(15) not null,

foreign key(id\_pets) references Pets(id\_pets) on delete cascade,

foreign key(id\_doctor) references Doctors(id\_doctor) on delete cascade

);

1. Creación de la tabla Procedures:

create table Procedures(

id\_procedure int(15) not null auto\_increment primary key,

procedure\_type varchar(255) not null,

date1 date not null,

description1 varchar(255) not null,

inputs varchar(255) not null,

recovery\_time varchar(255) not null,

id\_pets int(15) not null,

foreign key(id\_pets) references Pets(id\_pets) on delete cascade

);

1. Creación de la tabla Vaccine\_History:

create table Vaccine\_History(

id\_vaccine int(15) not null auto\_increment primary key,

name2 varchar(255) not null,

lot int(15),

application\_date date not null,

next\_dose date not null,

id\_pets int(15) not null,

foreign key(id\_pets) references Pets(id\_pets) on delete cascade

);

1. Creación de la tabla Health\_History:

create table Health\_history(

id\_history int(15) not null auto\_increment primary key,

weight varchar(255) not null,

allergies varchar(255) not null,

conditions varchar(255) not null,

behavior varchar(255) not null,

id\_pets int(15) not null,

id\_procedure int(15) not null,

id\_vaccine int(15) not null,

foreign key(id\_pets) references Pets(id\_pets) on delete cascade,

foreign key(id\_procedure) references Procedures(id\_procedure) on delete cascade,

foreign key(id\_vaccine) references Vaccine\_History(id\_vaccine) on delete cascade

);

1. Creación de la tabla Appointments:

create table Appointments(

id\_appointment int(15) not null auto\_increment primary key,

date1 date not null,

hour1 varchar(255) not null,

state varchar(255) not null,

process1 varchar(255) not null,

reason varchar(255) not null,

id\_owners int(15) not null,

foreign key(id\_owners) references Owners (id\_owners) on delete cascade

);

1. Creación de la tabla Medicines:

create table Medicines(

id\_medicine int(15) not null auto\_increment primary key,

name1 varchar(255) not null,

type1 varchar(255) not null,

manufacturer varchar(255) not null,

quantity int(15) not null,

expiration\_date date not null,

price int(15) not null

);

1. Creación de la tabla Products:

create table Products(

id\_product int(15) not null auto\_increment primary key,

type1 varchar(255) not null,

manufacturer varchar(255) not null,

quantity int(15) not null,

expiration\_date date not null,

price int(15) not null

);

1. Creación de la tabla Inventary:

create table Inventary(

id\_inventory int(15) not null auto\_increment primary key,

supplier varchar(255) not null,

id\_medicine int(15) not null,

id\_product int(15) not null,

foreign key(id\_medicine) references Medicines(id\_medicine) on delete cascade,

foreign key(id\_product) references Products(id\_product) on delete cascade

);

1. Creación de la tabla Administrator:

create table Administrator(

id\_administrator int(15) not null auto\_increment primary key,

name1 varchar(255) not null,

phone int(15) not null,

email varchar(255) not null,

nit int(15) not null,

password1 varchar(255) not null

);

1. Creación de la tabla Details:

create table Details(

id\_details int(15) not null auto\_increment primary key,

unit\_value int(15) not null,

subtotal int(15) not null,

tax int(15) not null,

invoice\_date date not null,

id\_consult int(15) not null,

foreign key(id\_consult) references Consult(id\_consult) on delete cascade

);

1. Creación de la tabla Invoice:

create table Invoice(

id\_invoice int(15) not null auto\_increment primary key,

total int(15) not null,

cufe varchar(255) not null,

code1 varchar(255) not null,

id\_owners int(15) not null,

id\_administrator int(15) not null,

id\_details int(15) not null,

foreign key(id\_owners) references Owners(id\_owners) on delete cascade,

foreign key(id\_administrator) references Administrator(id\_administrator) on delete cascade,

foreign key(id\_details) references Details(id\_details) on delete cascade

);

1. Ceación de la tabla Adoption:

create table Adoption(

id\_adoption int(15) not null auto\_increment primary key,

full\_name varchar(255) not null,

phone int(15) not null,

email varchar(255) not null,

address varchar(255) not null,

adoption\_date date not null,

id\_administrator int(15) not null,

id\_owners int(15) not null,

foreign key(id\_administrator) references Administrator(id\_administrator) on delete cascade,

foreign key(id\_owners) references Owners(id\_owners) on delete cascade

);

1. Creación de la tabla Visit\_History:

create table Visits\_History(

id\_visit int(15) not null auto\_increment primary key,

quantity int(15) not null,

profit varchar(255) null,

id\_owners int(15) not null,

foreign key(id\_owners) references Owners(id\_owners) on delete cascade

);

**Relaciones y Cardinalidades**

Se realizaron las relaciones y cardinalidades respectivas del modelo conceptual con sus entidades para tener mejor visualización de la base de datos.

* **Pets - Owners**
* **Relación: “**Tiene”.
* **Descripción:** Una mascota tiene varios dueños.
* **Cardinalidad: 1-N (**uno a muchos).
* **Pets - Vaccine\_History**
* **Relación:** “Tiene”.
* **Descripción:** Una mascota tiene un historial de vacunas.
* **Cardinalidad: 1-1** (uno a uno).
* **Pets - Consult**
* **Relación:** “Puede”.
* **Descripción:** Una mascota puede tener varias consultas.
* **Cardinalidad: 1-N** (uno a muchos).
* **Consult - Doctors**
* **Relación:** “Muchos”.
* **Descripción:** Muchas consultas pueden tener un doctor.
* **Cardinalidad: N-1** (muchos a uno).
* **Procedures - Pets**
* **Relación:** “Puede”.
* **Descripción:** Un procedimiento puede tener muchas mascotas.
* **Cardinalidad: 1-N** (uno a muchos).
* **Medicines - Inventary**
* **Relación:** “tiene”.
* **Descripción:** Un medicamento tiene un inventario.
* **Cardinalidad: 1-1** (uno a uno).
* **Appointments - Owners**
* **Relación:** “Muchos”.
* **Descripción:** Muchas citas pueden tener un dueño.
* **Cardinalidad: N-1** (muchos a uno).
* **Administrator - Invoice**
* **Relación:** “Puede”.
* **Descripción:** Un administrador puede hacer muchas facturas.
* **Cardinalidad: 1-N** (uno a muchos).
* **Administrator - Medicines**
* **Relación: “**Puede”.
* **Descripción:** Un administrador puede manejar muchos medicamentos.
* **Cardinalidad: 1-N** (uno a muchos).
* **Administrator - Inventary**
* **Relación:** “Puede”.
* **Descripción:** Un administrador puede manejar muchos inventarios.
* **Cardinalidad: 1-N** (uno a muchos),
* **Health\_History - Pets**
* **Relación:** “Tiene”.
* **Descripción:** Un historial de salud tiene muchas mascotas.
* **Cardinalidad: 1-N** (uno a muchos).
* **Invoice - Details**
* **Relación:** “Tiene”.
* **Descripción:** Una factura tiene un detalle.
* **Cardinalidad: 1-1** (uno a uno).
* **Adoption - Owners**
* **Relación:** “Puede”.
* **Descripción:** Muchas adopciones pueden tener un dueño.
* **Cardinalidad: N-1** (muchos a uno).
* **Products - Invoice**
* **Relación:** “Tiene”.
* **Descripción:** Un producto tiene un inventario.
* **Cardinalidad: 1-1** (uno a uno).
* **Owners - Inovice**
* **Relación:** “Tiene”.
* **Descripción:** Un dueño tiene muchas facturas.
* **Cardinalidad: 1-N** (uno a muchos).
* **Administrator - Products**
* **Relación:** “Maneja”.
* **Descripción:** Un administrador maneja muchos productos.
* **Cardinalidad: 1-N** (uno a muchos).
* **Owners - Additional\_Services**
* **Relación:** “Quiere”.
* **Descripción:** Un dueño quiere muchos servicios adicionales.
* **Cardinalidad: 1-N** (uno a muchos).

1. **Clases realizadas en el proyecto**

* **Owners**
* **Administrator**
* **Doctors**

1. **Jframe realizados en el proyecto**

* **Owners**
* **Administrator**
* **Doctors**

1. **Controladores realizados en el proyecto**

* **Owners**
* **Administrator**
* **Doctors**

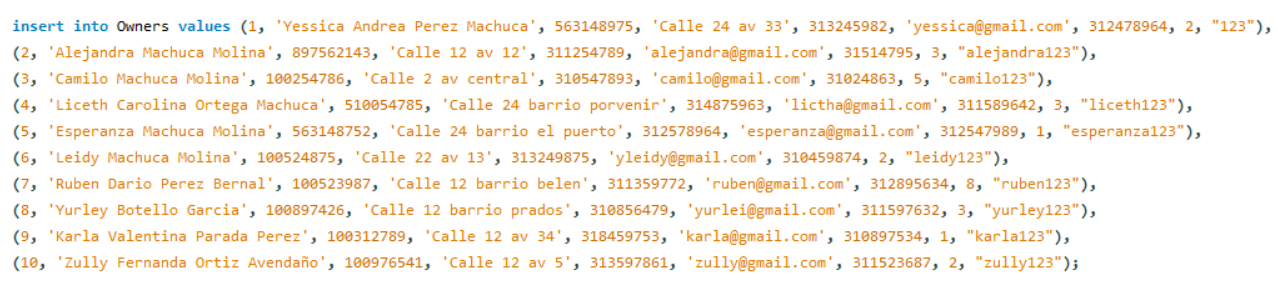
**Inserción de datos**

La inserción de datos en las tablas es una parte fundamental para la prueba de operatividad. Para insertar datos en una tabla específica, se utiliza la siguiente sintaxis

**Descripción**

Para aclarar el proceso de inserción de datos en la tablas de la Base de Datos de Happy Feet consideramos un ejemplo práctico:

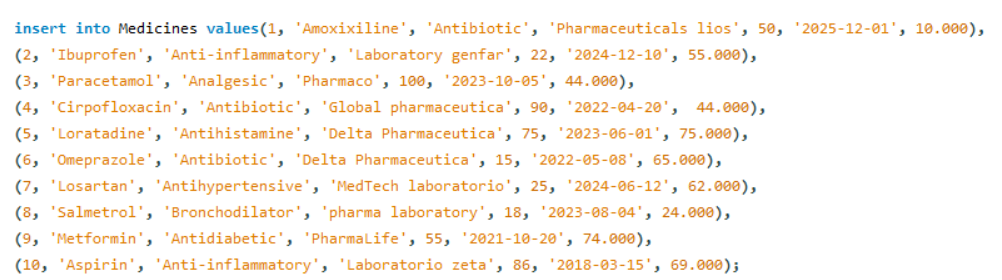
* Tabla Owners:



* Tabla Doctors:



* Tabla Medicines:



**Controladores**

El controlador es la parte que recibe las peticiones del usuario y coordina la respuesta de la aplicación.

1. Administrator:

* ControladorAdd.
* ControladorAdministrator.
* ControladorDelete.
* ControladorReport.
* ControladorUpdate.
* ControladorView.

1. Doctors:

* ControladorDoctors.
* ControladorProcedures.
* Controlador\_Vaccine.
* Controlador\_add.
* Controlador\_update.
* Controlador\_view.

1. Owners:

* ControladorAdd.
* ControladorOwners.
* ControladorUpdate.
* ControladorView.

**Modelos**

Es la parte de la aplicación que define los datos que contendrá.

1. Administrator:

* Conexión.
* Consultas.
* Pets.
* Procedures.
* Vaccine\_History.

1. Doctors:

* Conexión.
* Consultas.
* Doctors.
* Pets.
* Procedures.
* Vaccine\_History.

1. Owners:

* Additional\_Services.
* Appointments.
* Conexión.
* Consultas.
* Owners.
* Pets.
* Visits\_History.

**Vistas**

Es la representación visual de los datos y la interfaz del usuario (UI) de una aplicación.

1. Administrator:

* Vista\_Administrator.
* Vista\_Delete.
* Vista\_Login.
* Vista\_add.
* Vista\_menu.
* Vista\_reports.
* Vista\_update.
* Vista\_view.

1. Doctors:

* Vista\_Procedures.
* Vista\_Vaccine.
* Vista\_add.
* Vista\_login.
* Vista\_menu.
* Vista\_update.
* Vista\_view.

1. Owners:

* Add.
* FrameU.
* Menu.
* Update.
* View.

**Clases**

1. Administrator:

* AdditionalServices.
* Administrator.
* Adoption.
* Doctors.
* Inventory.
* Medicines.
* Owners.
* Pets.
* PetsAttended.
* Products.
* Vaccine\_History.
* VeterinarianPerformence.

1. Doctors:

* Doctors.
* Pets.
* Procedures.
* Vaccine\_History.

1. Owners:

* Consultas.
* Owners.
* Pets.
* Visits\_History.