Universidad Autónoma del Estado de Hidalgo

Proyecto final

Base datos distribuida para veterinaria

Integrantes:

Daniel Monroy Garnica
Saul Jimenez Mercado
Alexis Ortiz Jaen

1. Introducción

En la actualidad, las empresas que operan en diferentes regiones enfrentan retos considerables en la administración de datos. Esto es especialmente relevante para una clínica veterinaria que tiene varias sedes en diversas áreas del estado. La correcta administración de la información es esencial para ofrecer un servicio de calidad a los clientes, lo cual implica tener los registros de animales, propietarios, trabajadores, tratamientos, servicios y productos actualizados. La adopción de un sistema de bases de datos distribuidas podría ser la alternativa más adecuada para enfrentar estos problemas. Este tipo de estructura permite que la información sea guardada y gestionada en varios sitios, lo que posibilita un acceso rápido y coherente a los datos desde cualquier lugar dentro de la red. Asimismo, una base de datos distribuida puede diseñarse para ser escalable, lo que implica que puede ajustarse de manera sencilla al crecimiento de la empresa sin perjudicar su rendimiento. La meta principal de este proyecto es crear y establecer una base de datos distribuida que no solo optimice la gestión de la información en la clínica veterinaria, sino que también mejore la experiencia del cliente al asegurar que los servicios sean personalizados y eficaces.

Planteamiento del problema

La veterinaria enfrenta desafíos en la gestión de información debido a la dispersión geográfica de sus establecimientos. Esto puede generar problemas de acceso a datos, duplicidad de información y falta de consistencia en la atención al cliente. La implementación de una base de datos distribuida resolvería estos problemas al permitir un acceso rápido y uniforme a la información en todas las sucursales.

Objetivo general

Diseñar e implementar una base de datos distribuida que permita una gestión eficiente y homogénea de la información en todos los establecimientos de la veterinaria.

Objetivos específicos

- Identificar y definir las necesidades de la veterinaria para establecer qué tipos de datos deben ser gestionados y cómo deben ser accesibles.
- Establecer una estructura adecuada que permita la distribución de los datos en diferentes ubicaciones físicas y garantice la integridad, disponibilidad y seguridad de la información.
- Asegurar que sea capaz de sincronizar la información de manera eficiente entre todos los establecimientos de la veterinaria, garantizando que los datos se actualicen y estén disponibles de forma homogénea en todas las ubicaciones.

Alcances

- Implementar en todas las sucursales de la veterinaria las bases de datos para que se pueda tener información de las mascotas sin importar a la sucursal a la que se acuda.
- Crear bases de datos homogéneas para que el uso en cualquier sucursal sea el mismo.
- Utilizar réplicas para tener la información lo más eficientemente posible.

limites

No se va a poder hacer una escalabilidad vertical ya que no se cuenta con el presupuesto necesario para obtener los recursos

2. Marco teórico

Metodologia de analisis

Los diagramas de casos de uso muestran el comportamiento esperado del sistema. No muestran el orden en que se realizan los pasos. La definición del límite de sistema determina lo que se considera externo o interno del sistema.Un actor representa un rol que desempeña un objeto externo. Un objeto puede desempeñar varios roles y, por lo tanto, está representado por varios actores.
Una asociación ilustra la participación del actor en el caso de uso. Una caso de uso es un conjunto de eventos que se produce cuando un actor usa un sistema para completar un proceso. Normalmente, un caso de uso es un proceso relativamente grande, no un paso individual o una transacción.
Una historia de usuario es la unidad de trabajo más pequeña en un marco ágil. Es un objetivo final, no una función, expresado desde la perspectiva del usuario del software.
Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente.
El propósito de una historia de usuario es articular cómo un elemento de trabajo entregará un valor particular al cliente. Hay que tener en cuenta que los "clientes" no tienen porqué ser usuarios finales externos en el sentido tradicional, también pueden ser clientes internos o colegas dentro de la organización que dependen del equipo. Las historias de usuario son unas pocas frases en lenguaje sencillo que describen el resultado deseado. No entran en detalles, ya que los requisitos se añaden más tarde, una vez acordados por el equipo.
El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software.
Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.
UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos.

Metodología de diseño

La estrategia Top-down se integra de un conjunto de pasos que guían el proceso de diseño de un sistema de Base de Datos distribuida. Este método implica comenzar con una visión general del sistema o proyecto y luego descomponerlo en componentes más pequeños y detallados. Comienza con una comprensión completa del sistema, incluyendo sus objetivos y requisitos generales, para luego dividirlo en partes manejables y analizar cada una de ellas de manera detallada. A medida que se avanza en el diseño, se refina y ajusta cada componente para asegurar que se cumplan los requisitos del sistema completo. Se aplica en áreas como el diseño de sistemas electrónicos complejos, redes de datos, controladores lógicos y diseño asistido por computadora (CAD), ayudando a manejar ensambles complejos de manera eficiente y asegurando que todos los componentes trabajen juntos para cumplir con los objetivos generales del sistema.

Metodología de desarrollo

El desarrollo en cascada es un procedimiento lineal que se caracteriza por dividir los procesos de desarrollo en sucesivas fases de proyecto. Al contrario que en los modelos iterativos, cada una de estas fases se ejecuta tan solo una vez. Los resultados de cada una de las fases sirven como hipótesis de partida para la siguiente. El modelo de cascada se utiliza, especialmente, en el desarrollo de software.
Se compone de las siguientes fases:

1. Análisis
   Todo proyecto de software comienza con una fase de análisis que incluye un estudio de viabilidad y una definición de los requisitos. En el estudio de viabilidad se evalúan los costes, la rentabilidad y la factibilidad del proyecto de software. El estudio de viabilidad da como resultado un pliego de

condiciones, un plan y una estimación financiera del proyecto, así como una propuesta para el cliente, si fuera necesario. A continuación, se realiza una definición detallada de los requisitos, incluyendo un análisis de la situación de salida y un concepto. Mientras que los análisis de salida se encargan de describir la problemática en sí, el concepto ha de definir qué funciones y características debe ofrecer el producto de software para cumplir con las correspondientes exigencias. La definición de los requisitos da como resultado un pliego de condiciones, una descripción detallada de cómo se han de cumplir los requisitos del proyecto, así como un plan para la prueba de aceptación, entre otros. Por último, la primera fase del waterfall model incluye un análisis de la definición de los requisitos en el que los problemas complejos se dividen en pequeñas tareas secundarias y se elaboran las correspondientes estrategias de resolución.

2. Diseño
   La fase de diseño sirve para formular una solución específica en base a las exigencias, tareas y estrategias definidas en la fase anterior. En esta fase, los desarrolladores de software se encargan de diseñar la arquitectura de software, así como un plan de diseño detallado del mismo, centrándose en componentes concretos, como interfaces, entornos de trabajo o bibliotecas. La fase de diseño da como resultado un borrador preliminar con el plan de diseño del software, así como planes de prueba para los diferentes componentes.

3. Implementación
   La arquitectura de software concebida en la fase de diseño se ejecuta en la fase de implementación, en la que se incluye la programación del software, la búsqueda de errores y las pruebas unitarias. En la fase de implementación, el proyecto de software se traduce al correspondiente lenguaje de programación. Los diversos componentes se desarrollan por separado, se comprueban a través de las pruebas unitarias y se integran poco a poco en el producto final. La fase de implementación da como resultado un producto de software que se comprueba por primera vez como producto final en la siguiente fase (prueba alfa).

4. Prueba
   La fase de prueba incluye la integración del software en el entorno seleccionado. Por norma general, los productos de software se envían en primer lugar a los usuarios finales seleccionados en versión beta (pruebas beta). Las pruebas de aceptación desarrolladas en la fase de análisis permiten determinar si el software cumple con las exigencias definidas con anterioridad. Aquellos productos de software que superan con éxito las pruebas beta están listos para su lanzamiento.

5. Mantenimiento
   Una vez que la fase de prueba ha concluido con éxito, se autoriza la aplicación productiva del software. La última fase del modelo en cascada incluye la entrega, el mantenimiento y la mejora del software.

3. Análisis de requerimientos

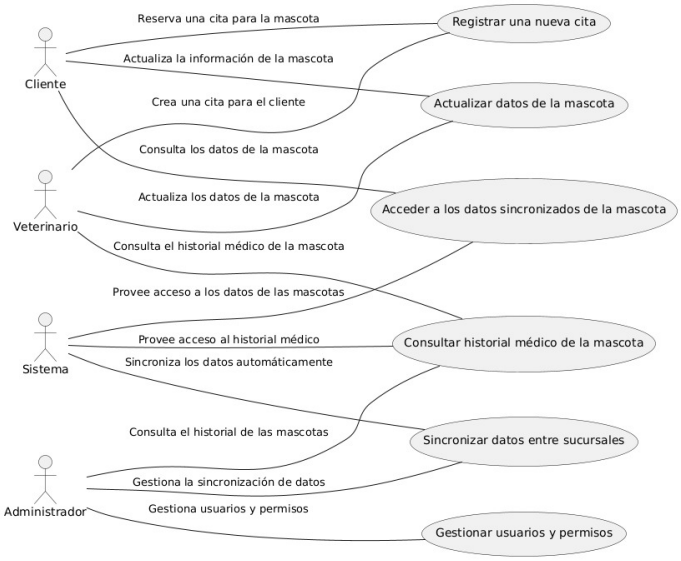
En este sistema se busca solucionar la relación que hay entre diferentes entidades que están relacionadas con la veterinaria como lo son:

1. Mascota
2. Dueño
3. Veterinario
4. Tratamiento
5. Servicio
6. Alimento
7. Cosmético
8. Sucursal

9. Cita
10. consulta
11. venta

A continuación se muestran casos de uso para demostrar el funcionamiento general del sistema.

1	Manejo de Base de Datos para Veterinaria		
Descripción	Este caso de uso se centra en cómo se maneja la base de datos creada para una veterinaria que cuenta con varias sucursales en distintas partes del estado. Los empleados de la veterinaria utilizarán el sistema para registrar y consultar información sobre clientes, mascotas, empleados y productos, todo con el fin de mejorar la eficiencia en la gestión de datos y asegurar un servicio de calidad.		
Actores	• Dueño de la mascota • Empleado de la veterinaria • Administrador del sistema • Sistema de base de datos		
Pre condiciones	• La base de datos debe estar en funcionamiento y ser accesible desde todas las sucursales. • Los actores deben tener los permisos necesarios para acceder a la información relevante. • Es esencial contar con una conexión de red estable para asegurar la sincronización de datos entre las réplicas.		
Post condiciones	• Los datos se almacenan y replican correctamente en toda la base de datos distribuida. • Se garantiza un acceso eficiente y seguro a la información.		
Secuencia Normal	**#**	**Acción (actor)**	**Reacción (sistema)**
	1	El empleado inicia sesión en el sistema con sus credenciales.	El sistema verifica las credenciales y asigna los permisos adecuados.
	2	El empleado consulta o registra información sobre una mascota, su dueño, un tratamiento, un servicio o un producto.	La base de datos procesa la solicitud y accede a la información almacenada en las réplicas para ofrecer una respuesta más rápida.
	3	Si el empleado registra una nueva cita, servicio o producto.	La información se almacena y replica automáticamente en la base de datos distribuida.
	4	El sistema confirma la acción y muestra el resultado al usuario.	
Flujo Alternativo	**#**	**Situación**	**Acción del sistema**
	2.1	Si la autenticación falla	El sistema rechaza el acceso y muestra un mensaje de error
	2.2	Si un usuario sin permisos intenta acceder a información restringida.	El sistema deniega el acceso.
	2.3	En caso de fallo en la red.	El sistema almacena temporalmente la solicitud y la sincroniza cuando la conexión está disponible.
Excepciones	**#**	**Acción (actor)**	**Reacción (sistema)**
	p	En el caso de que una transacción falle debido a problemas de conectividad.	El sistema deberá reintentar la acción una vez restablecida la conexión.
	q	En caso de inconsistencias en los datos replicados.	Se aplicarán mecanismos de resolución de conflictos automatizados.
Rendimiento	El sistema debe llevar a cabo las acciones mencionadas en los pasos 1 a 4 en un tiempo máximo de 2 segundos.		
Frecuencia	Se espera que este caso de uso se realice alrededor de 500 veces al día.		
Importancia	Crucial		
Urgencia	Inmediata		
Comentarios	Este caso de uso es esencial para el funcionamiento adecuado de la veterinaria, garantizando que la información esté disponible y segura en todas sus sucursales.		



Después vimos como va a interactuar el personal y los dueños con el sistema y que es lo que se puede hacer

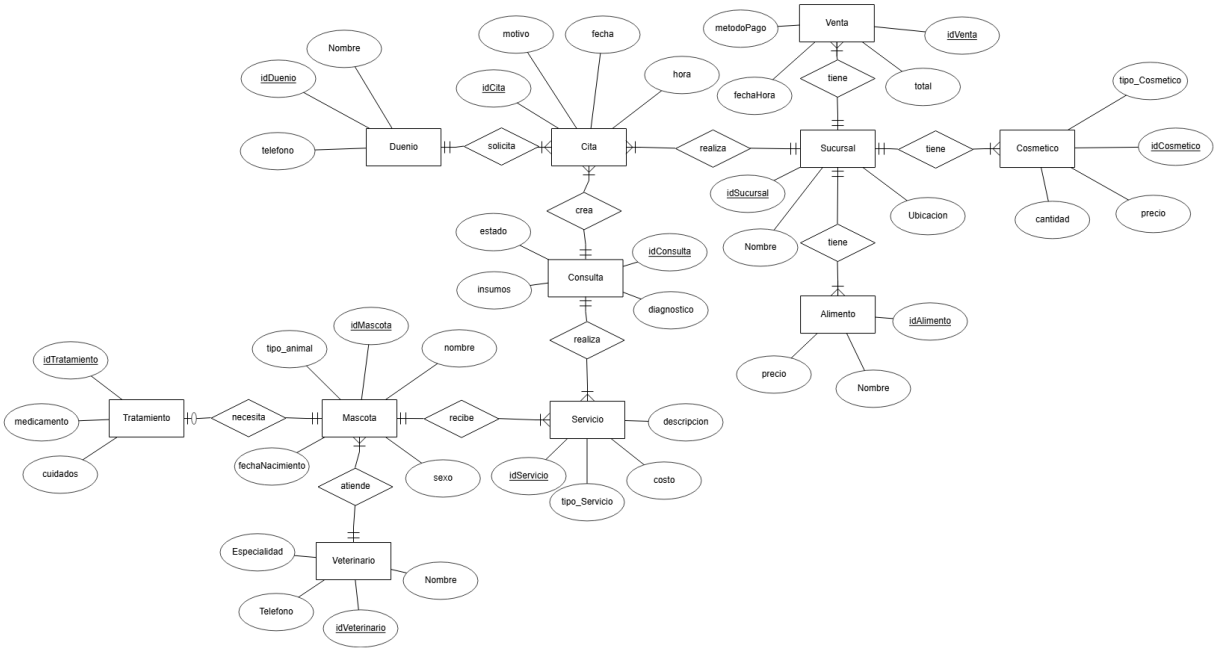
1	Administración del Sistema por Administrador		
Descripción	El administrador podrá gestionar los usuarios del sistema, configurar permisos y mantener la seguridad de la información.		
Actores	• Administrador del sistema • Sistema de base de datos		
Pre condiciones	• El administrador debe contar con credenciales de acceso de nivel administrativo.		
Post condiciones	• Se gestionan correctamente los usuarios y permisos. • Se mantiene la integridad y seguridad del sistema.		
Secuencia Normal	**#**	**Acción (actor)**	**Reacción (sistema)**
	1	El administrador inicia sesión en el sistema.	El sistema valida las credenciales y otorga acceso a la administración.
	2	El administrador gestiona usuarios y permisos.	El sistema almacena los cambios y confirma la acción.
	3	El administrador revisa registros de actividad y seguridad.	Se muestra el historial de accesos y eventos relevantes.
Excepciones	**#**	**Acción (actor)**	**Reacción (sistema)**
	p	Si el administrador intenta eliminar un usuario activo.	Se muestra una advertencia y se requiere confirmación.
	q	Si hay un intento de acceso no autorizado.	Se registra el evento y se envía una alerta.
Rendimiento	El sistema debe llevar a cabo las acciones mencionadas en los pasos 1 a 4 en un tiempo máximo de 2 segundos.		
Frecuencia	Cada caso de uso se espera que se lleve a cabo múltiples veces al día dependiendo del actor.		
Importancia	Vital		
Urgencia	Inmediatamente		
Comentarios	Cada actor tiene roles y permisos específicos dentro del sistema, asegurando la seguridad y eficiencia en la gestión de la veterinaria.		

1	Registro y Consulta de Información por Dueño de Mascota	
Descripción	El dueño de una mascota podrá acceder al sistema para consultar información sobre su mascota, historial de citas y tratamientos, así como registrar nuevas citas.	
Actores	• Dueño de la mascota • Sistema de base de datos	
Pre condiciones	• El usuario debe estar registrado en el sistema. • La mascota debe estar vinculada al dueño en la base de datos.	
Post condiciones	• Se muestra la información solicitada. • Se registra correctamente una nueva cita si fue solicitada.	
Secuencia Normal	**#**	**Acción (actor)**	**Reacción (sistema)**
	1	El dueño accede al sistema con sus credenciales.	El sistema valida el acceso.
	2	El dueño consulta la información de su mascota.	Se muestra el historial de citas y tratamientos.
	3	El dueño solicita una nueva cita.	El sistema registra la cita y envía una confirmación.
Excepciones	**#**	**Acción (actor)**	**Reacción (sistema)**
	p	Si las credenciales son incorrectas.	Se deniega el acceso y muestra un mensaje de error.
	q	Si la mascota no está registrada.	Se informa al usuario que debe contactar a la veterinaria.
Rendimiento	El sistema debe llevar a cabo las acciones mencionadas en los pasos 1 a 4 en un tiempo máximo de 2 segundos.	
Frecuencia	Cada caso de uso se espera que se lleve a cabo múltiples veces al día dependiendo del actor.	
Importancia	Vital	
Urgencia	Inmediatamente	
Comentarios	Cada actor tiene roles y permisos específicos dentro del sistema, asegurando la seguridad y eficiencia en la gestión de la veterinaria.	

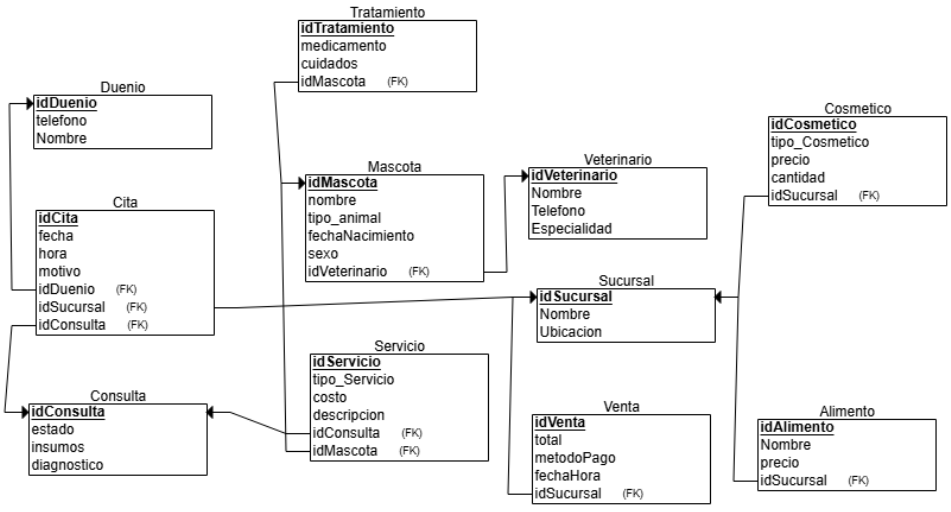
1	Gestión de Información por Empleado de la Veterinaria	
Descripción	El empleado de la veterinaria podrá registrar, actualizar y consultar información sobre mascotas, dueños, tratamientos y servicios ofrecidos.	
Actores	• Empleado de la veterinaria • Sistema de base de datos	
Pre condiciones	• El empleado debe estar registrado y autenticado en el sistema.	
Post condiciones	• Se almacenan correctamente los datos registrados o modificados. • Se muestra la información consultada.	
Secuencia Normal	**#**	**Acción (actor)**	**Reacción (sistema)**
	1	El empleado inicia sesión en el sistema.	El sistema valida el acceso y muestra las opciones disponibles.
	2	El empleado consulta información de una mascota o servicio.	Se muestran los datos almacenados.
	3	El empleado registra o actualiza información.	El sistema guarda los cambios y confirma la acción.
Excepciones	**#**	**Acción (actor)**	**Reacción (sistema)**
	p	Si el empleado intenta modificar datos sin permisos.	Se deniega la acción y muestra un mensaje de error.
	q	Si la base de datos no está disponible.	Se notifica el error y se recomienda intentar más tarde.
Rendimiento	El sistema debe llevar a cabo las acciones mencionadas en los pasos 1 a 4 en un tiempo máximo de 2 segundos.	
Frecuencia	Cada caso de uso se espera que se lleve a cabo múltiples veces al día dependiendo del actor.	
Importancia	Vital	
Urgencia	Inmediatamente	
Comentarios	Cada actor tiene roles y permisos específicos dentro del sistema, asegurando la seguridad y eficiencia en la gestión de la veterinaria.	

Diseño

Modelo Entidad-Relacion



Modelo Relacional



Desarrollo

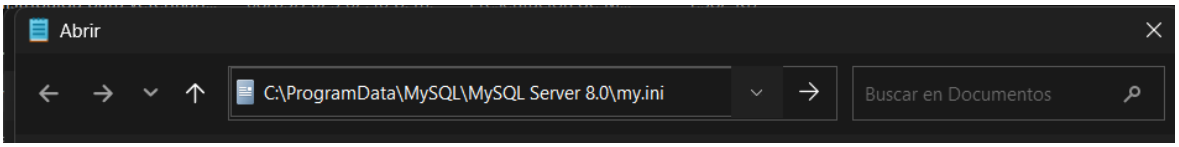
Lo primero fue crear las tablas necesarias para la base de datos las cuales se crearon con el siguiente código:

```
CREATE TABLE Duenio (
    idDuenio INT NOT NULL,
    telefono VARCHAR(20) NOT NULL,
    various phone number formats
    nombre VARCHAR(100) NOT NULL,
    PRIMARY KEY (idDuenio)
);


CREATE TABLE Sucursal (
    idSucursal INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    ubicacion VARCHAR(255) NOT NULL,
    PRIMARY KEY (idSucursal)
);

CREATE TABLE Veterinario (
    idVeterinario INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    telefono VARCHAR(20) NOT NULL,
    especialidad VARCHAR(100) NOT NULL,
    PRIMARY KEY (idVeterinario)
);

CREATE TABLE Cosmetico (
    idCosmetico INT NOT NULL,
    tipo_Cosmetico VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    cantidad INT NOT NULL,
    idSucursal INT NOT NULL,
    PRIMARY KEY (idCosmetico),
    FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal)
);

CREATE TABLE Alimento (
    idAlimento INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    idSucursal INT NOT NULL,
    PRIMARY KEY (idAlimento),
    FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal)
);

CREATE TABLE Consulta (
    idConsulta INT NOT NULL,
    estado VARCHAR(50) NOT NULL,
    insumos VARCHAR(255) NOT NULL,
    diagnostico VARCHAR(250) NOT NULL,
    PRIMARY KEY (idConsulta)
);
```

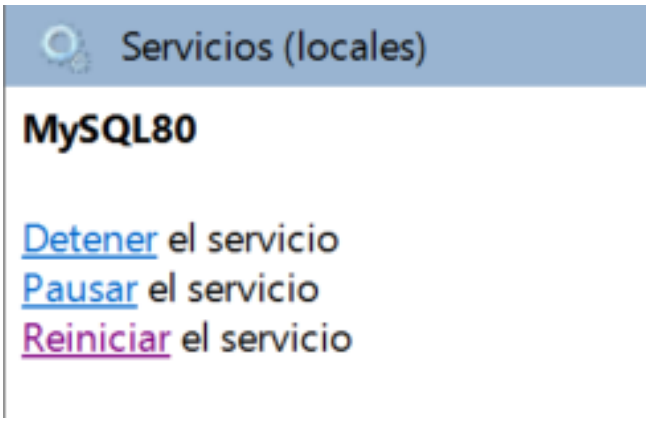
```
51
52 CREATE TABLE Venta (
53     idVenta INT NOT NULL,
54     total DECIMAL(10, 2) NOT NULL,
55     metodoPago VARCHAR(50) NOT NULL,
56     fechaHora DATETIME NOT NULL,
57     idSucursal INT NOT NULL,
58     PRIMARY KEY (idVenta),
59     FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal)
60 );
61
62 CREATE TABLE Mascota (
63     idMascota INT NOT NULL,
64     nombre VARCHAR(100) NOT NULL,
65     tipo_animal VARCHAR(100) NOT NULL,
66     fechaNacimiento DATE NOT NULL,
67     sexo VARCHAR(20) NOT NULL,
68     idVeterinario INT NOT NULL,
69     PRIMARY KEY (idMascota),
70     FOREIGN KEY (idVeterinario) REFERENCES Veterinario(idVeterinario)
71 );
72
73 CREATE TABLE Tratamiento (
74     idTratamiento INT NOT NULL,
75     medicamento VARCHAR(255) NOT NULL,
76     cuidados VARCHAR(255) NOT NULL,
77     idMascota INT NOT NULL,
78     PRIMARY KEY (idTratamiento),
79     FOREIGN KEY (idMascota) REFERENCES Mascota(idMascota)
80 );
81
82 CREATE TABLE Cita (
83     idCita INT NOT NULL,
84     fecha DATE NOT NULL,
85     hora TIME NOT NULL,
86     motivo VARCHAR(250) NOT NULL,
87     idDuenio INT NOT NULL,
88     idSucursal INT NOT NULL,
89     idConsulta INT NOT NULL,
90     PRIMARY KEY (idCita),
91     FOREIGN KEY (idDuenio) REFERENCES Duenio(idDuenio),
92     FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal),
93     FOREIGN KEY (idConsulta) REFERENCES Consulta(idConsulta)
94 );
95
96 CREATE TABLE Servicio (
97     idServicio INT NOT NULL,
98     tipo_Servicio VARCHAR(100) NOT NULL,
99     costo DECIMAL(10, 2) NOT NULL,
100    descripcion VARCHAR(250) NOT NULL,
101    idConsulta INT NOT NULL,
102    idMascota INT NOT NULL,
103    PRIMARY KEY (idServicio),
104    FOREIGN KEY (idConsulta) REFERENCES Consulta(idConsulta),
```

```
105     FOREIGN KEY (idMascota) REFERENCES Mascota(idMascota)
106 );
```

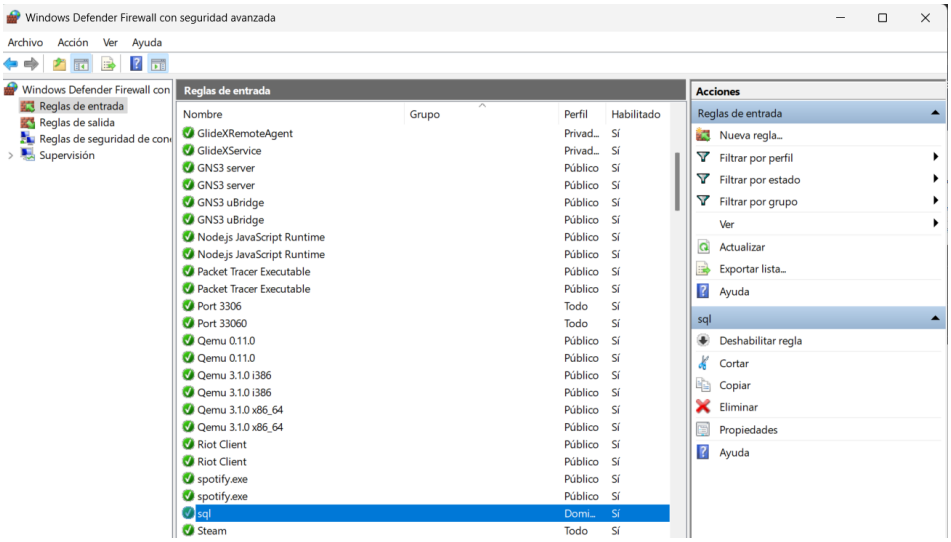
Después de eso tuvimos que hacer modificaciones en un archivo de MySQL llamado my.ini para que se puedan aceptar conexiones desde cualquier IP. Se tiene que modificar desde el bloc de notas y poner bind-address = 0.0.0.0 después se guardan los cambios y se reinicia el servicio de MySQL 80.




```
# SERVER SECTION
# ----------------------------------------------------------------------
#
# The following options will be read by the MySQL Server. Make sure that
# you have installed the server correctly (see above) so it reads this
# file.
#
[mysqld]
bind-address = 0.0.0.0
```



Con esta parte ya hecha se debe de crear una regla en el firewall con la que la computadora va a permitir que se pueda manejar todas las bases de datos de forma remota y así que sea de manera mas segura las conexiones.

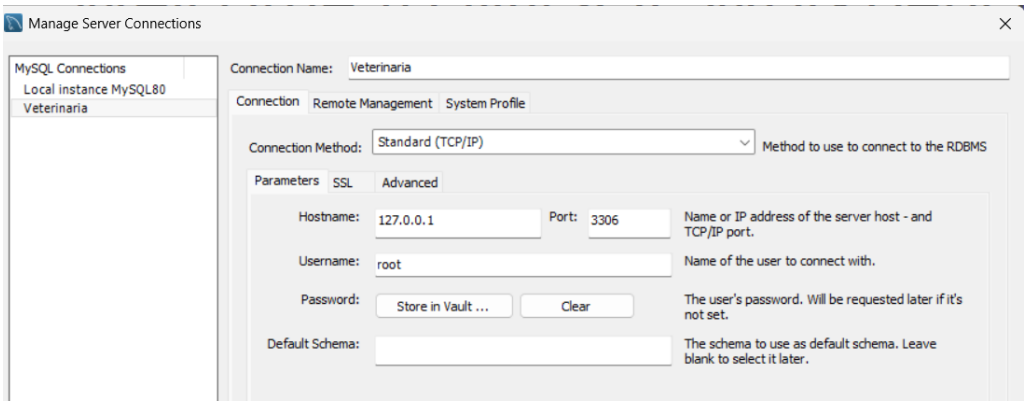


Por medio de un modem realizamos red LAN a la que van a estar conectadas las computadoras

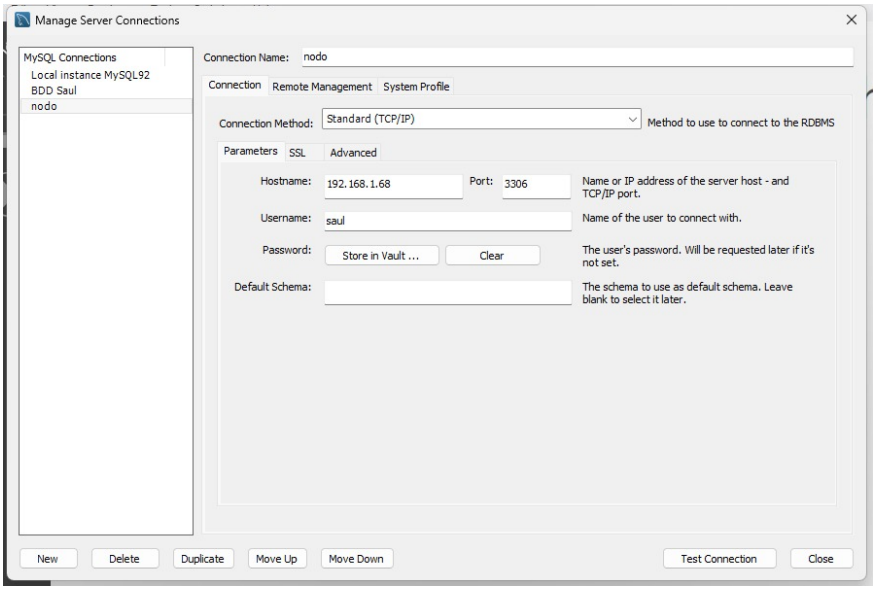
Luego verificamos que ip tiene la computadora para saber como configurar el host.

```
Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.68
```

Luego en MySQL ponemos los datos para poder acceder, la dirección IP se pone por defecto ya que es en la que vamos a trabajar aunque la dirección IP de la computadora sea diferente. Este es la configuración que aparece para el host de la base de datos.



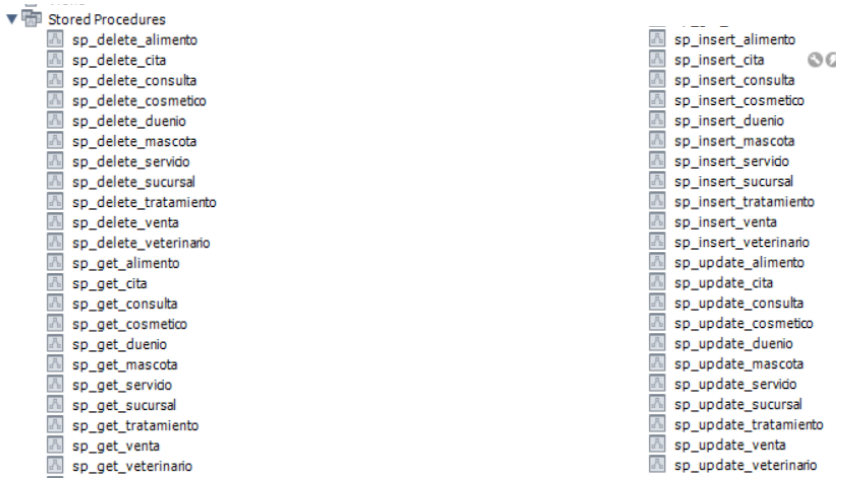
Para la otra computadora la configuración si debe de llevar la dirección IP del host para saber con que computadora va a estar manejando la base de datos.



Y luego se da los permisos por medio de un comando en sql.

```
CREATE USER 'saul'@'192.168.1.64' IDENTIFIED BY 'admin123';
GRANT ALL PRIVILEGES ON *.* TO 'saul'@'192.168.1.64' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

También creamos varios procedimientos almacenados y triggers para el funcionamiento de la base de datos y que fuera mas eficiente.

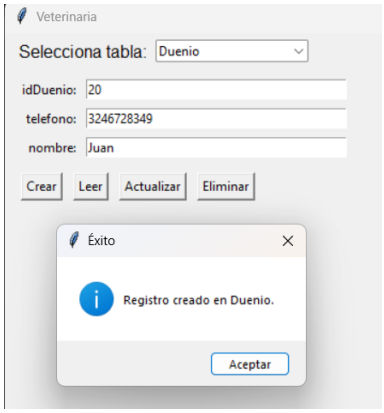


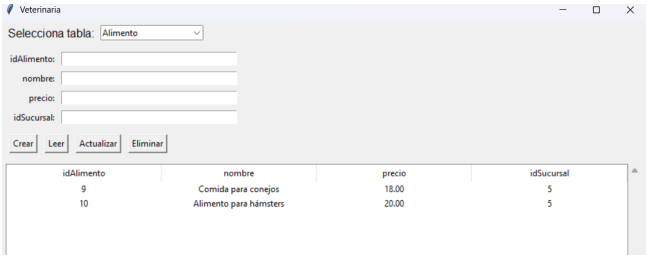
Por ahora eso seria todo para mysql, ahora vamos a juntarlo con una interfaz en python la cual va a tener que hacer una conexión por medio de la librería mysql-connector-python y asi poder usar todas las tablas, los procedimientos almacenados y triggers dentro de la base de datos. Una cosa importante tambien es que en python se debe de especificar la IP, el usuario, la contraseña y la base de datos usados y así poder modificar la base de datos.

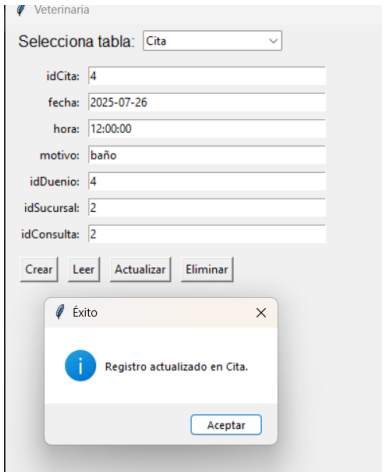
```python
DB_CONFIG = {
    'host': '127.0.0.1',
    'user': 'daniel',
    'password': 'admin123',
    'database': 'veterinaria'
}
```

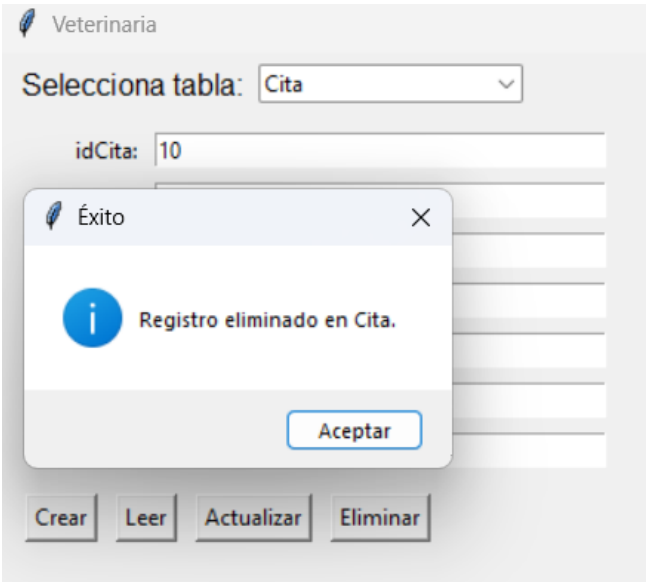
Interfaces

Para las interfaces creamos 2, una para los administradores los cuales van a poder hacer las funciones de CRUD para todas las tablas, mientras que la otra interfaz sera para los clientes y puedan realizar citas y revisar susu citas pasadas.









Aquí se muestra una de los cambios que se hicieron en ambos nodos.

Result Grid | Filter Rows:

idDuenio	telefono	nombre
1	555-1234	Juan Pérez
2	555-5678	María López
3	555-8765	Carlos García
4	555-4321	Ana Martínez
5	555-1111	Luis Fernández
6	555-2222	Sofía Rodríguez
7	555-3333	Pedro Sánchez
8	555-4444	Laura Gómez
9	555-5555	Jorge Díaz
10	555-6666	Claudia Torres
11	7717183...	Francisco
12	843749	macario
15	23091301	se
20	3246728...	Juan
NULL	NULL	NULL

Conclusiones

En conclusión, la creación de una base de datos distribuida no es para nada sencillo ya que se deben de tener bastantes conocimientos acerca de como crear una estructura de una base de datos y también hay una gran dificultad a la hora de poder juntar una aplicación de Python con MySQL. Algunas de las mejoras que consideramos es crear mas interfaces y también dar mejores métodos de seguridad.

Referencias Bibliográficas

References

[1] Crear un diagrama de casos de uso UML - Soporte técnico de Microsoft.(**s.f.**). *https://support.microsoft.com/es-es/topic/crear-un-diagrama-de-casos-de-uso-uml-92cc948d-fc74-466c-9457-e82d62ee1298*

[2] Rehkopf, D. M.(**s.f.**). Historias de usuario — Ejemplos y plantilla — Atlassian. Atlassian. *https://www.atlassian.com/es/agile/project-management/user-stories*

[3] Qué es el lenguaje unificado de modelado (UML).(**s.f.**). Lucidchart. *https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml*

[4] Cornejo Velázquez, E., & Clavel Maqueda, M. (**2023, 13 octubre**). Estrategia de implementación Top-Down [Diapositivas]. *https://repository.uaeh.edu.mx/bitstream/bitstream/handle/123456789/20482/top-down.pdf?sequence=1&isAllowed=y*

[5] Equipo editorial de IONOS. (**2019, 21 marzo**). El modelo en cascada: desarrollo secuencial de software. IONOS Digital Guide. *https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/*