



Centro Industrial y Desarrollo
Empresarial Soacha
Regional Cundinamarca



Manual Técnico

(Pets Heaven)

Julian Velez Quitian Kevin

Palacios Arce Cristian Arrieta

Romero Juana Violeta Espejo

**CENTRO INDUSTRIAL Y DESARROLLO EMPRESARIAL – CIDE ANÁLISIS Y DESARROLLO DE
SOFTWARE - ADSO
SOACHA, CUNDINAMARCA 2025**



1. Introducción
2. Objetivos del Proyecto
 - 2.1 Objetivo general
 - 2.2 Objetivos específicos
3. Alcance del proyecto
4. Personal Involucrado
 - 4.1 Diagramas de Casos de Uso
5. Características de los usuarios
6. Requerimientos
 - 6.1 Requerimientos funcionales
 - 6.2 Requerimientos no funcionales
7. Arquitectura del proyecto
 - 7.1 Definición metodología
 - 7.2 Ciclo de vida
 - 7.3 Diagrama de despliegue
8. Fundamentos y herramientas utilizadas
9. Requisitos del Sistema
10. Procesos del Software
11. Instalación de Aplicaciones
12. Diagrama de Clases
13. Modelo entidad Relación
14. Diagrama entidad relación
15. Diccionario de Datos
16. Glosario
17. Referencias



Control de versiones

FECHA	VERSIÓN	DESCRIPCION	AUTOR
20/06/2025	1.0	Primera versión del manual técnico del sistema de gestión veterinaria Pets Heaven.	Julian Velez Quitian Estudiante de Análisis y desarrollo de software, Servicio nacional de aprendizaje. Tel: 3134853945 Correo: julivelez337@gmail.com

Tabla 1. Control de versiones del manual técnico 1.

introducción

El presente documento servirá para identificar las técnicas usadas para el desarrollo de este aplicativo web, que se dará con el fin de explicar y facilitar el entendimiento de los procesos tales como, registrar usuarios, administradores, veterinarios y mascotas, así mismo el sistema de roles, agendamiento e historiales médicos con los que el sistema cuenta, dando explicaciones claras para que el personal que tenga contacto con él pueda darle el uso correspondiente y eficiente.

Así mismo tener en cuenta que esto no debe ser tomado como una guía de aprendizaje sino como una guía para documentar las diferentes técnicas usadas para la creación de este proyecto, adicional leer detenidamente los pasos descritos para la debida instalación del sistema el cual debe tener en cuenta que el usuario contara con conocimientos básicos en el uso de equipos tales como computadores y sus periféricos.

Además, se debe tener en cuenta la secuencia de usos o implementaciones de herramientas tecnológicas (React, Emailjs, Supabase, Express), Dependencias (librerías de react, boxicons, supabase, emailjs, sweetalert, framer-motion, fullcalendar, axios, compresorjs en el front end y cors, express, mysql, express-rate-limit, jsonwebtoken, bcrypt, dotenv,



cookie-parser en el back end.), Apis (BackEnd) consumibles para poder generar los llamados específicos.

2. Objetivos del proyecto

2.1. Objetivo general

Desarrollar un sistema web de gestión veterinaria que automatice los procesos administrativos y clínicos, con el fin de incrementar la productividad de la clínica y la satisfacción del cliente.

2.2. Objetivos específicos

- Fase de análisis

- Realizar el levantamiento de los requerimientos del software.
- Establecer requisitos funcionales y no funcionales.
- Identificación de riesgos.
- Definición de arquitectura.

- Fase de diseño

- Diseño de la arquitectura.
- Diagramas UML.
- Definición de tecnologías.

-Fase de implementación y despliegue

- Codificación según el diseño.
- Uso de control de versiones.
- Configuración de servidores.

3. Alcance del proyecto

El proyecto está destinado para servir a veterinarias de todo Soacha con más de 500 clientes mensuales, el sistema ofrecerá servicios en tiempo real para el agendamiento de



citas, usuarios, etc. además de incorporar roles para identificar a cada usuario dentro del sistema.

4. Personal involucrado

Nombre	Rol	Categoría profesional	Responsabilidad	Contacto	Apro v
Julian velez	Encargado de manuales y documenta ción.	Tecnólogo en análisis y desarrollo de software	Redacción de documentación pertinente	Tel: 3134853945 Correo: julivelez337@gmail.com	si
Kevin Palacios	Arquitecto de software.	Tecnólogo en análisis y desarrollo de software	Responsable de la accesibilidad del programa	Tel: 3114675099 Correo: kevin_spalacios@gmail.com	si
Cristian Arrieta	Analista de datos, Desarrollad or full stack.	Tecnólogo en análisis y desarrollo de software	Responsable de la seguridad, y la infraestructura del software.	Tel: 3205554483 Correo: cristianarrieta04@gmail .com	si
Juana Violeta	Desarrollad or FrontEnd.	Tecnólogo en análisis y desarrollo de software	Responsable del diseño web.	Tel: 3224523961 Correo: juanavioleta1325@gmail .com	si

Tabla 2. Encargado principal del desarrollo del aplicativo



5. Características de los usuarios

Tabla 3. Características de Cliente

TIPO DE USUARIO	Cliente
FORMACIÓN	Persona natural con conocimientos básicos en el uso de aplicación web, móviles o de escritorio.
HABILDADES	Conocimiento en el manejo de aplicaciones web
ACTIVIDADES	<ul style="list-style-type: none">-Navegar en el sitio web-Registrarse/logearse-Reestablecer contraseña-Ver historial de su mascota-Agendar, reprogramar, cancelar citas médicas en agenda personal-Actualizar foto de perfil de su mascota-Actualizar información de perfil propio

Tabla 3. Características de Administrador

TIPO DE USUARIO	Administrador
FORMACIÓN	Formación técnica para desarrollar las funciones de administración del sistema de información.
HABILDADES	<ul style="list-style-type: none">-Conocimiento extenso en el manejo de aplicaciones web, móviles o de escritorio.- Capacidad de resolución rápida de problemas.- Capacidad de comunicación oral y escrita.- Trabajo en equipo.
ACTIVIDADES	<ul style="list-style-type: none">-Navegar en el sitio web-Logearse-Reestablecer contraseña-Ver historial de mascotas-Agendar, reprogramar, cancelar citas médicas en agenda general y personal-CRUD de usuarios



	-CRUD de perfil propio -Asignar roles
--	--

Tabla 3. Características de usuario Personal

TIPO DE USUARIO	Personal
FORMACIÓN	Estudios centrados en ciencias y matemáticas, obtener una licenciatura, en ciencias biológicas, y completar una escuela de veterinaria.
HABILDADES	-Aplicar conocimientos -Afecto por los animales -Conocimiento en el manejo de aplicaciones web, móviles o de escritorio. - Capacidad de resolución rápida de problemas. - Capacidad de comunicación oral y escrita. - Trabajo en equipo. -Capacidad de adaptación
ACTIVIDADES	-Navegar en el sitio web -Logearse -Reestablecer contraseña -Ver historial de mascotas -Agendar, reprogramar, cancelar citas médicas en agenda personal -CRUD de perfil propio

5. Requerimientos

6.1 Requerimientos funcionales

- **Gestión de Usuarios:** El sistema debe permitir el registro y login de usuarios (clientes, veterinarios y administradores)
- **Gestión de Mascotas:** El sistema debe permitir registrar nuevas mascotas con datos como nombre, especie, raza, edad, propietario, etc.



- **Agendar citas:** El usuario debe poder agendar citas seleccionando fecha, hora, tipo de consulta y veterinario disponible.
El usuario debe poder agendar citas seleccionando fecha, hora, tipo de consulta y veterinario disponible.
- **Historial Médico de Mascotas:** El veterinario debe poder registrar diagnósticos, tratamientos y vacunas aplicadas.
- **Búsqueda de Mascotas o Citas:** El sistema debe permitir buscar mascotas por nombre o filtrar citas por fecha o veterinario.
- **Gestión de Servicios:** El sistema debe mostrar servicios para el cuidado de mascotas.
- **Panel de Administración:** El administrador debe poder gestionar usuarios, veterinarios, servicios y revisar reportes.

6.2 Requerimientos no funcionales

- **Usabilidad:** La interfaz debe ser intuitiva y fácil de usar para personas sin conocimientos técnicos.
- **Rendimiento:** El sistema debe responder a las acciones del usuario en menos de 2 segundos en condiciones normales.
- **Seguridad:** Las contraseñas deben almacenarse cifradas. Los usuarios deben tener roles para limitar accesos, Generar copias de seguridad cada semana.
- **Escalabilidad:** El sistema debe poder adaptarse para manejar un número creciente de mascotas, usuarios y citas.
- **Disponibilidad:** El sistema debe estar disponible el 99% del tiempo, salvo en mantenimiento programado.
- **Compatibilidad:** El sistema debe ser accesible desde navegadores y dispositivos móviles modernos.
- **Mantenibilidad:** El código debe estar documentado para facilitar futuras actualizaciones o correcciones.



-Accesibilidad: La página debe ser accesible a personas con discapacidades visuales, dando descripciones a objetos y facilitando la movilidad dentro de la página.

6. Arquitectura del proyecto

7.1 Definición metodología

La metodología que hemos adaptado en este proyecto es SCRUM ya que desde el inicio hemos usado historias épicas, y la competitividad entre desarrolladores así mejorando la efectividad y velocidad del desarrollo.

7.2 Ciclo de vida

El ciclo de vida que manejamos es:

-Análisis: En esta fase se identifican las necesidades del cliente o del proyecto. Se recopila y documenta la información necesaria para entender qué Problema se debe resolver. Aquí se definen los requisitos funcionales y no funcionales, y se establecen las bases para el desarrollo del sistema.

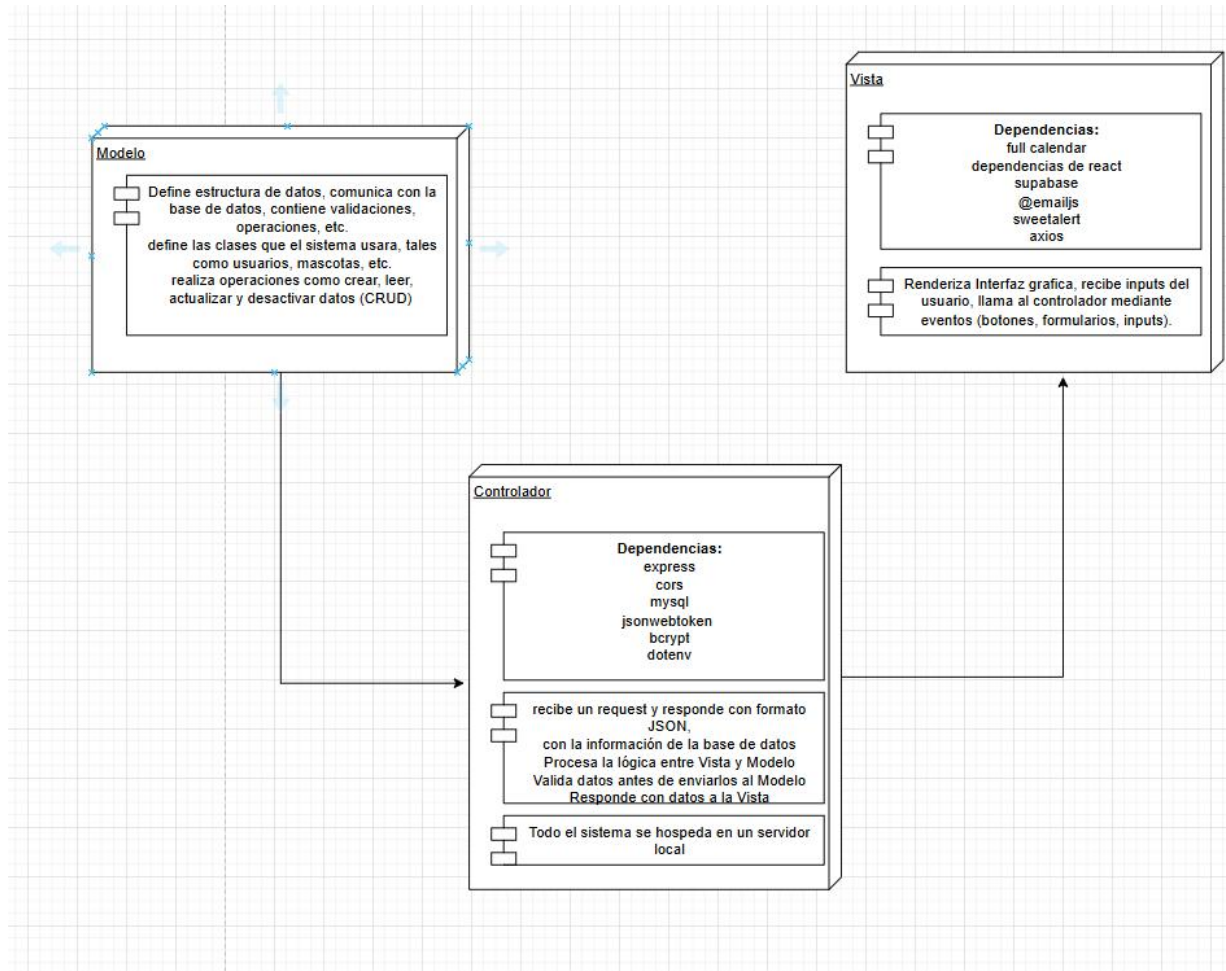
-Diseño: Con base en los requisitos definidos en el análisis, se planifica cómo se va a construir el sistema. Esto incluye la arquitectura del software, estructura de bases de datos, diseño de interfaces y flujo de la aplicación.

-Codificación: Es la fase donde se lleva a cabo la implementación del diseño usando en este caso Node, Express, React y mysql, css 3, HTML, Javasript. Se crean los módulos, componentes y funcionalidades del sistema según lo planificado.

-Despliegue: Una vez el sistema ha sido desarrollado y probado, se pone en marcha en el entorno real (producción). Aquí se aseguran aspectos como la configuración del servidor, base de datos y accesos del usuario.

-Mantenimiento: Después del despliegue, el sistema puede requerir mejoras, corrección de errores o adaptaciones a nuevos requerimientos

7.3 Diagrama de despliegue



Img.1 - Diagrama MVC. Autoría propia

Este sistema está basado en la arquitectura MODELO-VISTA-CONTROLADOR (MVC), separando responsabilidades para facilitar el mantenimiento y escalabilidad del proyecto.

-MODELO (base de datos): Esta sección es la encargada de definir la estructura de datos. Se comunica con la base de datos así mismo también define las clases que el sistema usara, y representa la lógica del negocio y la forma en la que debe comportarse.

-VISTA (front-end): Esta sección se encarga de mostrar la interfaz gráfica al usuario, recibiendo sus inputs a través de botones y formularios usando dependencias como full calendar, supabase, emailjs, etc con el fin de que la interfaz sea lo más interactiva y entendible posible.

-CONTROLADOR (back-end): Esta sección se encarga de recibir request y responderla por medio de un archivo de formato JSON con la información de la base de datos, procesa información entre el modelo



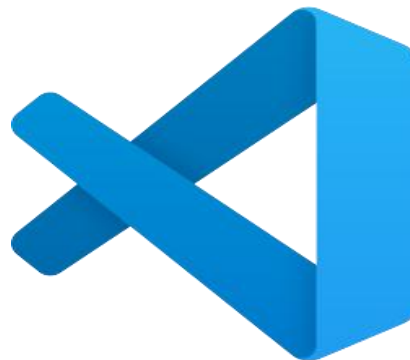
Manual Técnico

y la vista validando Dichos datos y dando respuesta en la vista.

8. Fundamentos y herramientas utilizadas

A continuación, se mostrará una lista de todas las herramientas usadas y necesarias para el despliegue y el debido funcionamiento del sistema.

-VISUAL STUDIO CODE:



Img 2- Logo visual studio code. (Wikipedia)

Visual studio code es un editor de código fuente ligero y a la vez eficaz, el cual fue usado para realizar enteramente el código del sistema, con ayuda de extensiones que el mismo programa da acceso.

-XAMPP:



Img 3- Logo Xampp. (draftdesignweb.com)

Xampp es una distribución de Apache, el cual se usó como servidor local para la base de datos.



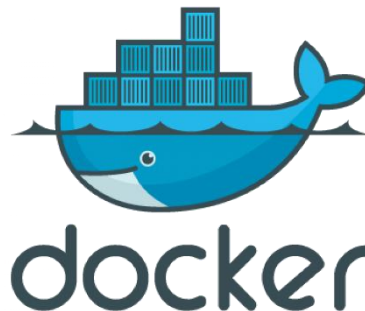
-SUPABASE:



Img 4- Logo de Supabase. (0hands.com)

Supabase es una plataforma de desarrollo de postgres, proporcionando alojamiento de datos, usamos esta plataforma para almacenar las imágenes que se suben por medio de los formularios y solo retornando la url que se almacena en la base de datos principal, usando para este fin el sistema de buckets (contenedor para archivos) implementado en supabase.

-DOCKER:

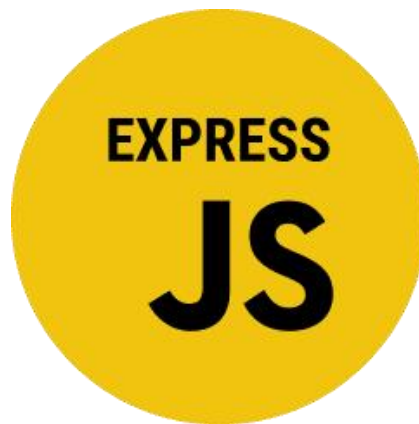


Img 5- Logo de docker. (Yo el programador)

Docker es una plataforma de software que permite a desarrolladores crear, probar e implementar aplicaciones de forma rápida, lo usamos con el fin de que nuestro sistema sea portable y multiplataforma.



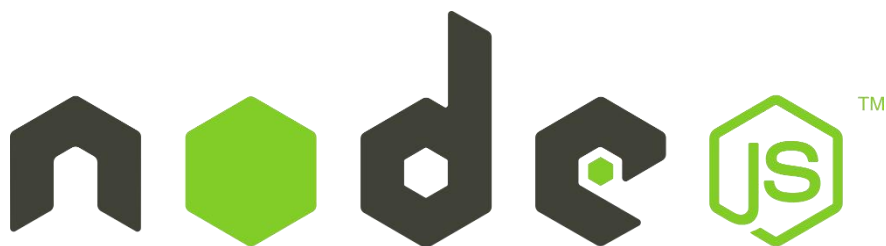
-EXPRESS:



Img 6- Logo Express. (ajeetchaulagain.com/)

Express es el framework web más popular de Node, y es a librería subyacente para un gran número de otros frameworks proporcionando mecanismo para la escritura de manejadores de peticiones con diferentes verbos (GET, POST, PUT, DELETE) HTTP.

-NODE.JS:



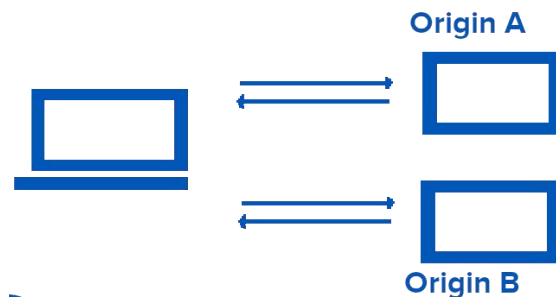
Img 7- Logo Node. ([Edureka.com](http://edureka.com))

Node.js es un entorno de ejecución JavaScript de código abierto y multiplataforma que se usa para desarrollar aplicaciones escalables de lado del servidor y la red.



Manual Técnico

-CORS:



Img 8- Imagen representativa de CORS. (agilicus.com)

Cors (Intercambio de recursos de origen cruzado) es un mecanismo basado en cabeceras HTTP que permite a un servidor indicar cualquier dominio, esquema o puerto con un origen distinto al suyo.

-MYSQL:



Img 9- Logo de MySql. (AWS.com)

MYSQL es un sistema de administración relacional de base de datos de código abierto que se usa para almacenar y gestionar datos usado por su fiabilidad, rendimiento, escalabilidad y facilidad de uso.



-JSON WEB TOKEN:



Img 10- Logo de JWT. (worldvectorlogo.com)

JWT (JSON Web Token) es un mecanismo para poder propagar entre dos partes y de forma segura la identidad de un determinado usuario.

-BCRYPT:



Img 11- Logo de Bcrypt. (github.com/topics/bcrypt-library)

Bcrypt es un algoritmo de hashing de contraseñas diseñado para protegerlas frente a ataques maliciosos, siendo diseñado intencionalmente lento y seguro.



Manual Técnico

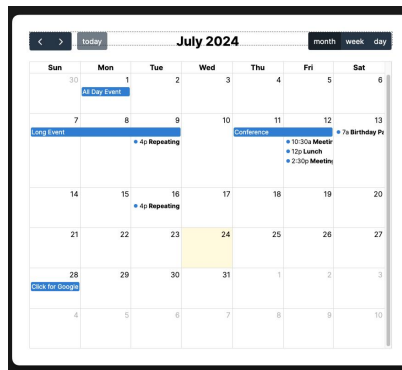
-DOTENV:



Img 12- Logo de Dotenv. (webcatalog.io)

Dotenv es un módulo sin dependencias, disponible a través de paquetes npm, que carga variables de entorno desde un archivo .env en nuestro caso.

-FULL CALENDAR:



Img 13- Representación de full calendar. (community.weweb.io)

Full calendar es una librería que trae por defecto una vista de un calendario con sus funcionalidades permitiendo representar datos procedentes de la base de datos para que sea en tiempo real y muy fácil y útil para todo tipo de usuario.



Manual Técnico

-DEPENDENCIAS DE REACT:

Las dependencias de React son elementos fundamentales que permiten que una aplicación se ejecute de manera fluida y sin problemas, entre las usadas tenemos:

react-dom

Librería que permite renderizar componentes de React en el DOM del navegador. Es el puente entre React y el HTML real.

react-hook-form

Librería para manejar formularios en React de forma simple y eficiente. Permite validaciones, manejo de errores e integración con inputs sin escribir tanto código.

react-router

Es una librería de enrutamiento para React que permite crear navegación entre diferentes páginas o componentes sin recargar la página. Define rutas (<Route>) y enlaces (<Link>) dentro de una aplicación SPA (Single Page Application).

-EMAILJS:



Img 14- Logo de Emailjs. (Emailjs.com)

Emailjs es una biblioteca ligera que ayuda a las aplicaciones a enviar correos electrónicos a través de un servidor SMTP, en este caso usado para enviar códigos de verificación de usuarios para el cambio de contraseña, y el registro en la página.

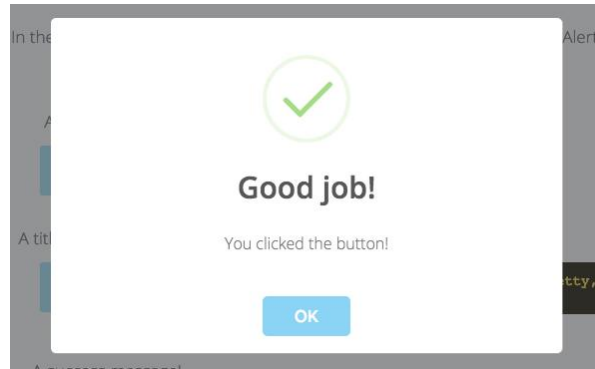


Centro Industrial y Desarrollo
Empresarial Soacha
Regional Cundinamarca



Manual Técnico

-SWEETALERT



Img 15- Representación de sweetalert. (Drupal.org)

Sweetalert es un complemento atractivo y responsivo para reemplazar las ventanas emergentes de JavaScript usado para generar alertas más llamativas y personalizadas.

- AXIOS:

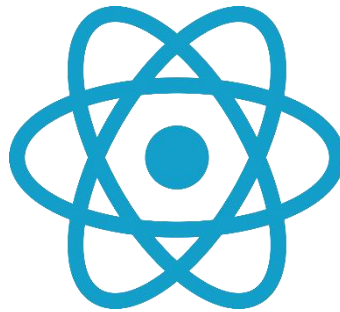


Img 16- Logo de Axios. (Axios.com)

Axios es un cliente HTTP basado en promesas para node.js y el navegador. Es isomorfo (puede ejecutarse en el navegador y en Node).



-REACT JS:



Img 17- Logo de React. (Wikipedia)

React es una biblioteca de JavaScript desarrollada para crear interfaces de usuario interactivas y modernas Especialmente para aplicaciones web basada en componentes.

-CSS:



Img 18- Logo de CSS. (JASodt.org)

CSS es un tipo de lenguaje que permite definir y crear la presentación visual de los documentos que ya se encuentran estructurados y escritos dentro del sistema de información en lenguaje HTML. En conclusión, CSS permite generar el diseño visual de las páginas web e interfaces de usuario.



Manual Técnico

9.Requisitos del Sistema

9.1 Lado del servidor: Software necesario (node.js, npm o yarn, base de datos MySQL) servidor web (puede ser local), sistema operativo compatible (Windows 10/11), Linux, macOS.

9.2 Lado del cliente: Navegador moderno (Chrome, FireFox, Edge, Safari, Google, Opera), JavaScript habilitado (navegadores lo tienen activado de por sí), conexión a internet estable, resolución de pantalla adecuada y un mínimo de 4 de RAM

NOTA: No necesita instalar nada ya que es un servicio web.

10.Procesos del Software

10.1 Procesos de entrada:

- Ingresa a la web (usuarios).
- Ingresa datos de registro o logeo (Administrador, personal y cliente).
- Ingresar datos para el registro de mascotas (Registros básicos en el sistema.).
- Ingresar datos para el registro de clientes (Registros básicos en el sistema.
- Ingresar datos para el registro de personal (Registros básicos en el sistema.).
- Ingresar datos para el agendamiento de una cita médica (Registros básicos en el sistema.).
- Ingresar datos para el ingreso de vacunas (Registros básicos en el sistema.).
- Ingresar datos para el ingreso de servicios (Registros básicos en el sistema.).
- Generar resultados de consulta (Factura, historial médico).

10.2 Procesos de salida:

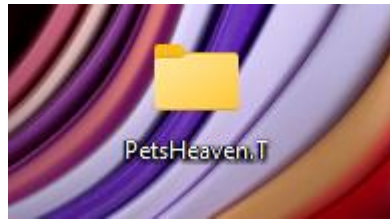
- Consulta de usuario (Usuarios).
- Consulta de personal (Personal).
- Consulta de categorías, servicios y categorías (Servicios, facturas).
- Consulta de citas (Citas).
- Consulta de facturas (Facturas).
- Consulta de vacunas (Vacunas).
- Descarga de facturas (Facturas).
- Descarga de historial médico (Historial médico.).

11.Instalación de Aplicaciones

Paso 1: Como primer paso tenemos que crear una carpeta con el nombre que se prefiera en este caso la llamaremos PetsHeaven.T:



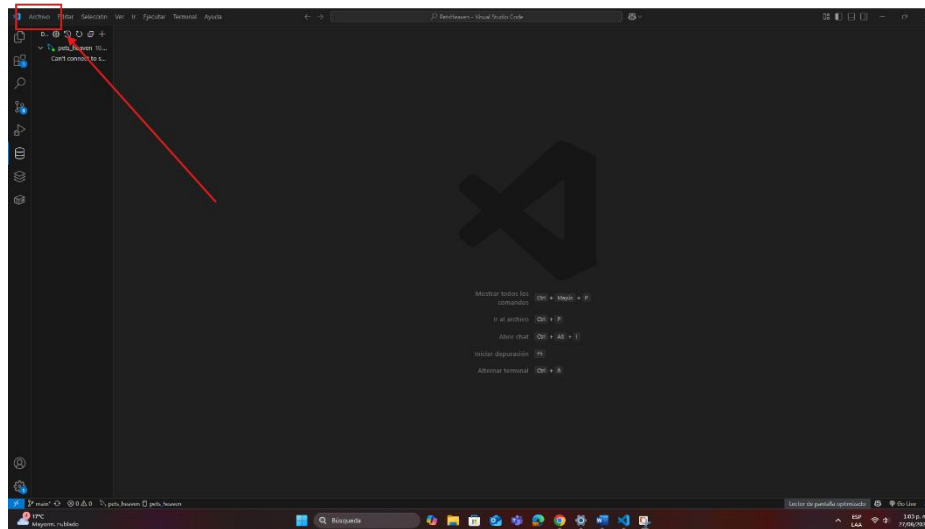
Manual Técnico



Img 19- Carpeta PetsHeaven.T creada. Autoría propia

Paso 2: En visual studio abrimos la carpeta que creamos antes:

2.1: nos dirigimos a la esquina superior izquierda donde dice ARCHIVOS:

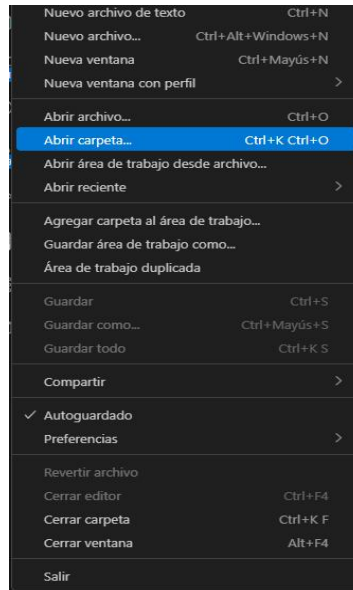


Img 20- Señalización del apartado 'archivos'. Autoría propia

2.2: Luego le damos al apartado ABRIR CARPETA:

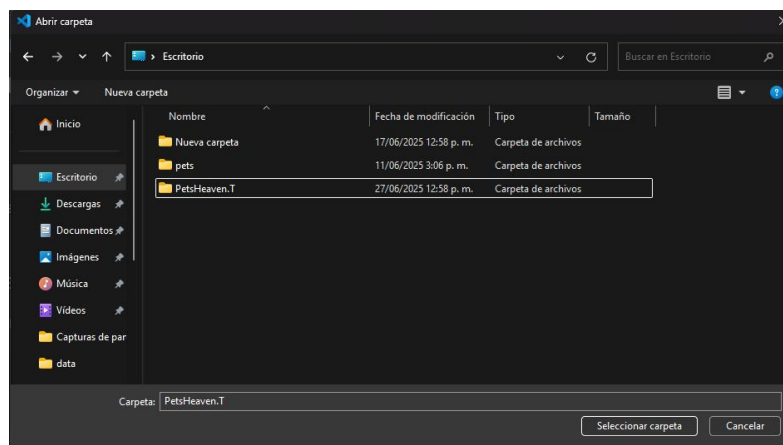


Manual Técnico



Img 21- Señalización del apartado 'abrir carpeta'. Autoría propia

2.3: Seleccionamos la carpeta anteriormente creada:



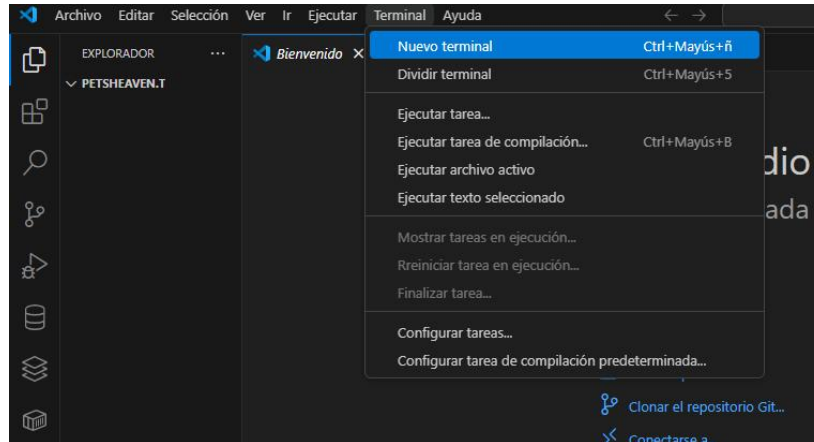
Img 22- Señalización de la carpeta a abrir. Autoría propia

Paso 3: Ya que tenemos la carpeta en nuestro visual studio es momento de clonar el repositorio de GitHub, a continuación, se explicará cómo, y se dejara el enlace del repositorio que es el siguiente: (<https://github.com/DICREY/PetsHeaven>)

3.1: Abrimos la terminal en visual studio usando ctr+ñ o el apartado TERMINAL y NUEVA TERMINAL en la Parte superior de la pantalla:



Manual Técnico



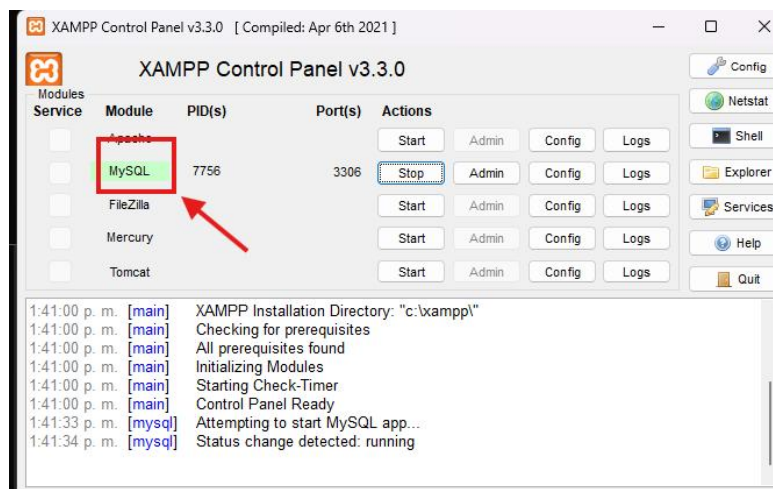
Img 23- Señalización del apartado 'nueva terminal'. Autoría propia

3.2: Clonamos el repositorio con el comando 'git clone (enlace)' en este caso el enlace que ya se proporcionó' Y seguidamente dar enter y esperar que el repositorio sea clonado:

```
PS C:\Users\Aprendiz\Desktop\PetsHeaven.T> git clone https://github.com/DICREY/PetsHeaven
Cloning into 'PetsHeaven'...
remote: Enumerating objects: 9977, done.
remote: Counting objects: 100% (1058/1058), done.
remote: Compressing objects: 100% (496/496), done.
remote: Total 9977 (delta 837), reused 734 (delta 553), pack-reused 8919 (from 1)
Receiving objects: 100% (9977/9977), 212.15 MiB | 92.00 KiB/s, done.
Resolving deltas: 100% (7121/7121), done.
```

Img 24- Confirmación de repositorio clonado. Autoría propia

3.3: A continuación, abrimos el programa XAMPP e iniciamos el módulo MySQL:



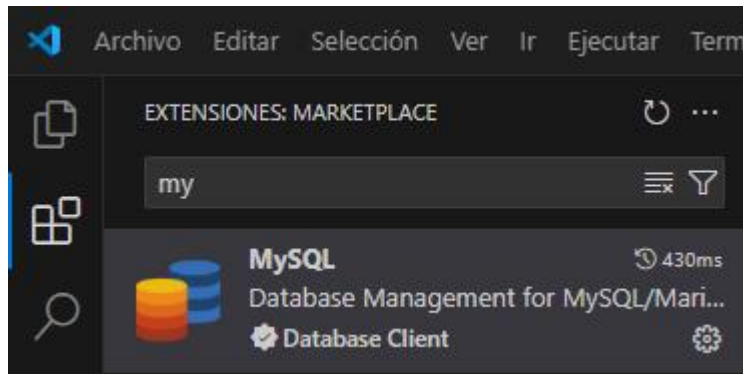
Img 25- Señalización del módulo a iniciar. Autoría propia



Manual Técnico

Paso 4: Ahora vamos a ver las extensiones necesarias para que todo corra como debe:

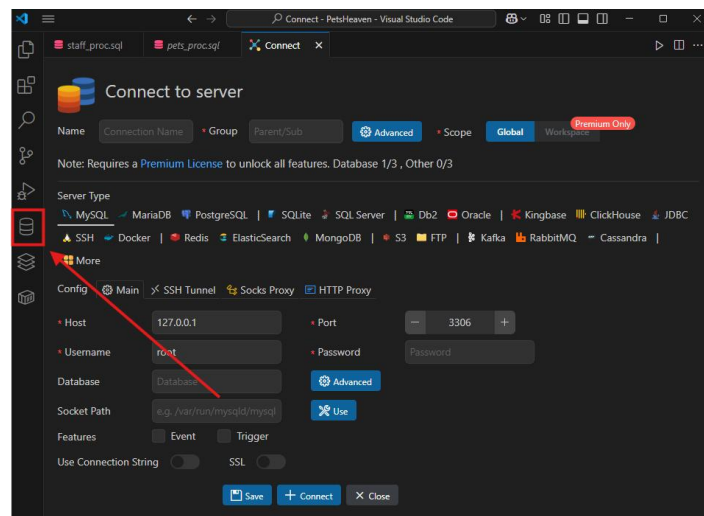
4.1: Instalar extensión MySQL en el apartado de extensiones:



Img 25- Extensión necesaria. Autoría propia

Paso 5: Ahora con el fin de que podamos crear la base de datos para que todo funcione debemos crear una Conexión a continuación se mostrará como:

5.1: Nos vamos al apartado DATABASE que gracias a la extensión que descargamos debe estar disponible en la barra lateral:



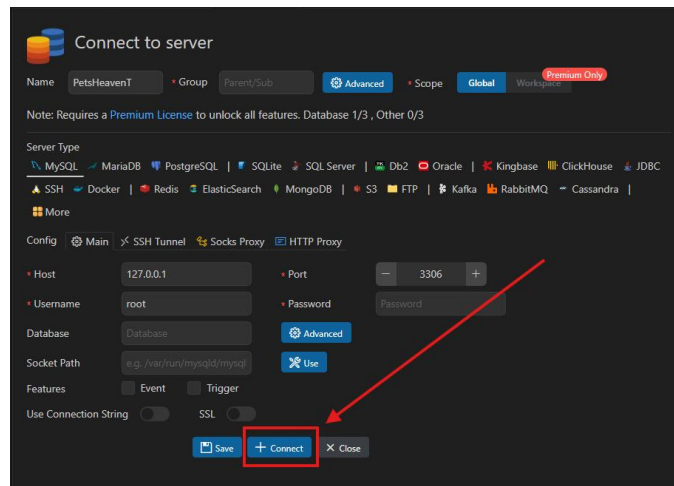
Img 26- Señalización del apartado 'database'. Autoría propia

5.2: Ahora solo debemos darle un nombre en este caso PetsHeavenT y conectar o poner contraseña si



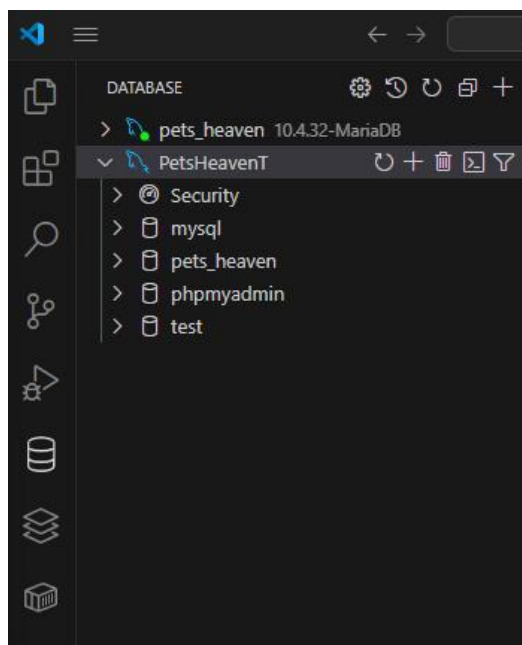
Manual Técnico

se tiene alguna:



Img 27- Señalización del apartado 'conectar'. Autoría propia

Paso 6: Ya con la conexión creada es momento de crear la base de datos desde la conexión antes mencionada, que debe aparecer en el mismo apartado en el que estamos ubicados:



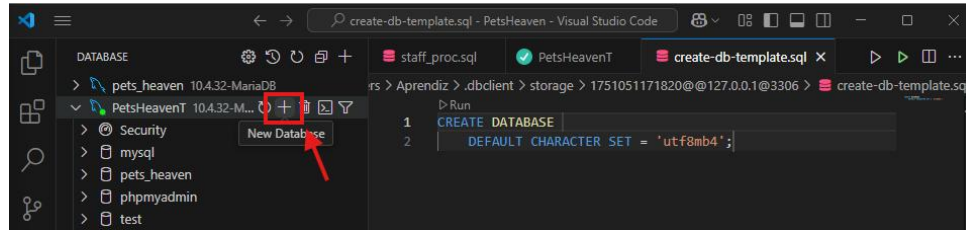
6.1: Ahora debemos seleccionar la conexión creada:

Img 28- Señalización del apartado 'conexión creada'. Autoría propia



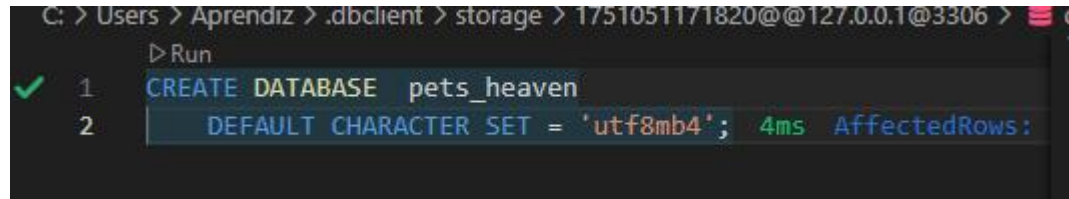
Manual Técnico

6.2: Ahora es momento de crear la base de datos dándole al signo más (+) que se muestra en la imagen:



Img 29- Señalización del apartado ‘nueva base de datos’. Autoría propia

6.3: Ahora se nos abre un archivo con un CREATE DATABASE en el cual debemos poner el nombre de la base de Datos (pets_heaven) y seguidamente correr el código:

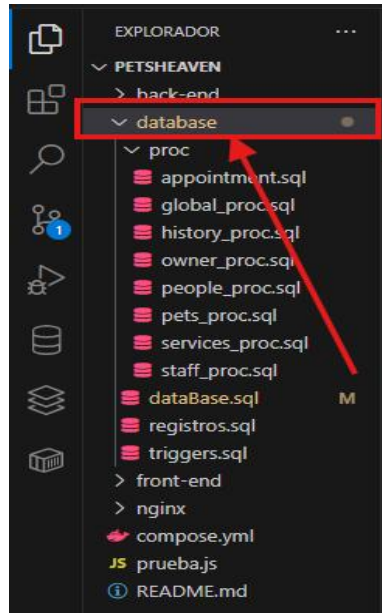


Img 30- Base de datos creada. Autoría propia

Paso 7: Bien ya tenemos la conexión creada junto con la base de datos, pero aun la base de datos no esta Funcionando completamente, para que todo este como debería estar debemos dirigirnos a la carpeta DATABASE del proyecto que clonamos anteriormente:



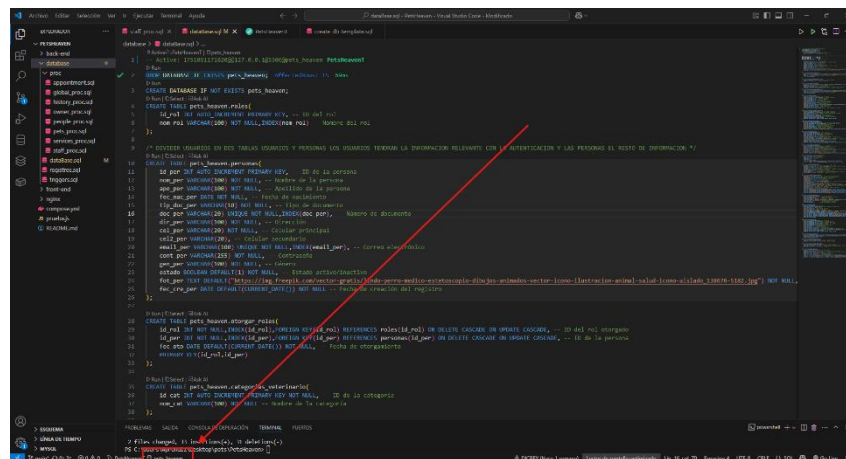
Manual Técnico



Img 31- Señalización de la carpeta database. Autoría propia

7.1: Ahora debemos entrar al archivo DataBase.sql y correr la base de datos seleccionándola como se mostrará:

7.1.1: Se selecciona la conexión en la parte inferior izquierda como se muestra:

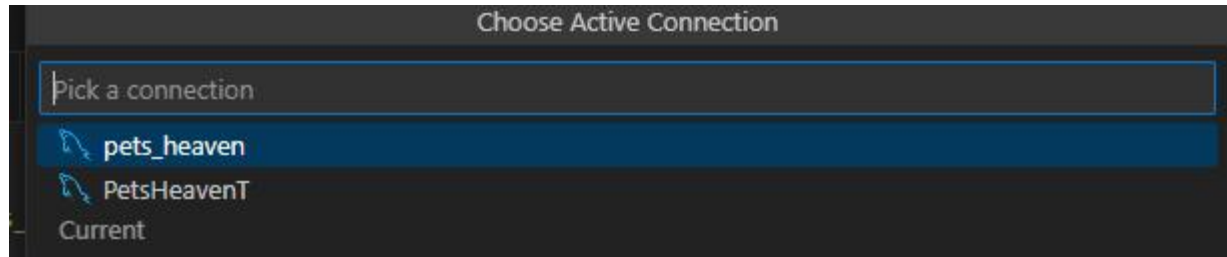


Img 32- Señalización del apartado de conexion. Autoría propia

7.1.2: Ahora se selecciona la conexión si es que tenemos más de una (en este caso se elige PetsHeavenT):

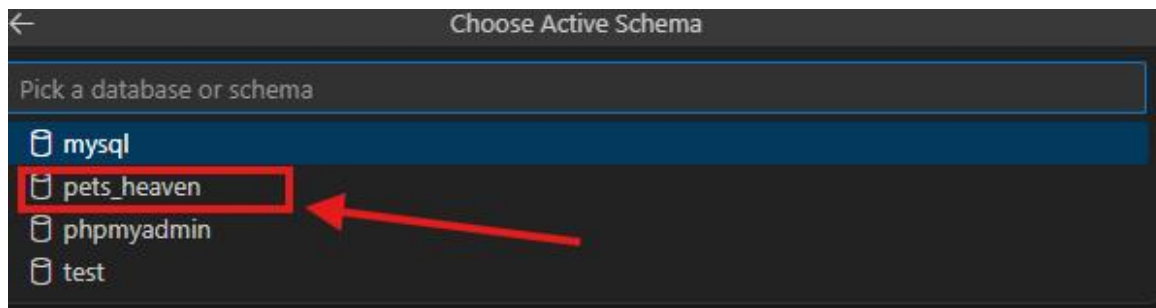


Manual Técnico



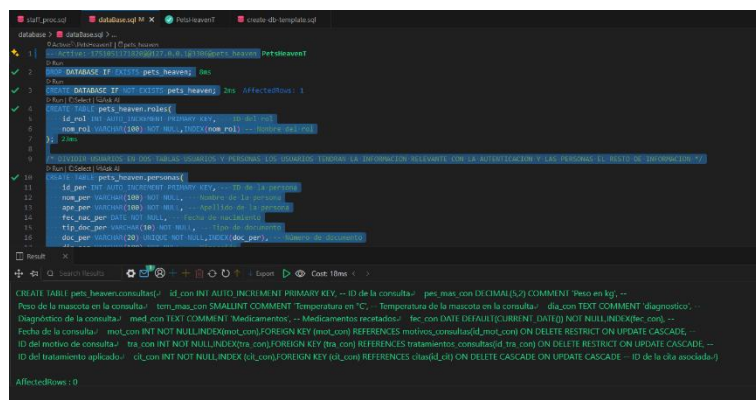
Img 33- Señalización del apartado 'conexiones'. Autoría propia

7.1.3: Una vez seleccionada la conexión se elige la base de datos que se va a usar en este caso Pets_Heaven:



Img 34- Señalización de la conexión a usar. Autoría propia

Paso 8: una vez seleccionado ahora solo debemos correr la base de datos, para más eficiencia podemos usar Los comandos crt+a para seleccionar todo el código y seguidamente crt+enter para correrlo todo:



Img 35- Base de datos creada. Autoría propia



Manual Técnico

NOTA: LA BASE DE DATOS ESTA DIVIDIDA EN VARIOS ARCHIVOS, LA BASE DE DATOS PRINCIPAL, LAS INSERCCIONES Y LOS PROCESOS ALAMCENADOS, POR ENDE, LOS PASOS DEL 7.1.1 HASTA EL PASO 8 SE DEBEN

REPETIR EN CADA ARCHIVO DE LA CARPETA DATABASE, INICIANDO CON LA BASE DE DATOS PRINCIPAL SIGUIDO

DE INSERCCIONES Y POR ÚLTIMO PROCESOS ALMACENADOS ESTOS ULTIMOS SIN ORDEN PARTICULAR.

Paso 9: Ahora necesitamos crear los archivos .env que no se clonan desde el repositorio por cuestiones de Seguridad:

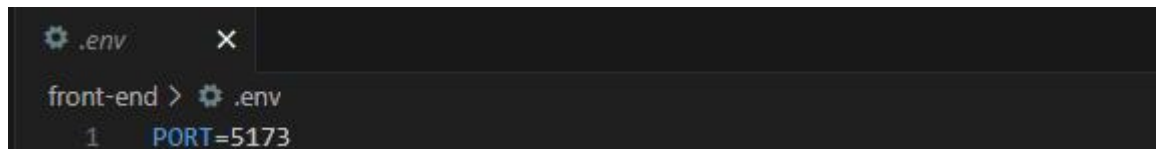
9.1: Creamos un archivo en el front-end llamado '.env':



Img 38- archivo '.env'. Autoría propia

Y en su interior ponemos:

PORT:5173



Img 39- Datos en el archivo '.env'. Autoría propia

9.2: Ahora hacemos los mismos procedimientos que en el paso 9.1 pero esta vez en el back-end:
Creamos archivo llamado '.env':



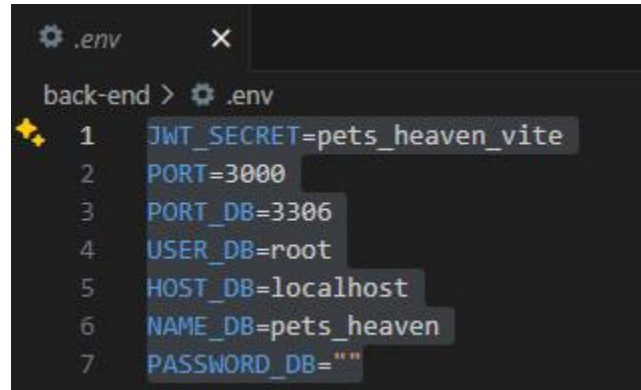
Img 40- archivo '.env'. Autoría propia

Y en su interior ponemos:

```
JWT_SECRET=pets_heaven_vite  
PORT=3000  
PORT_DB=3306  
USER_DB=root  
HOST_DB=localhost  
NAME_DB=pets_heaven  
PASSWORD_DB=""
```



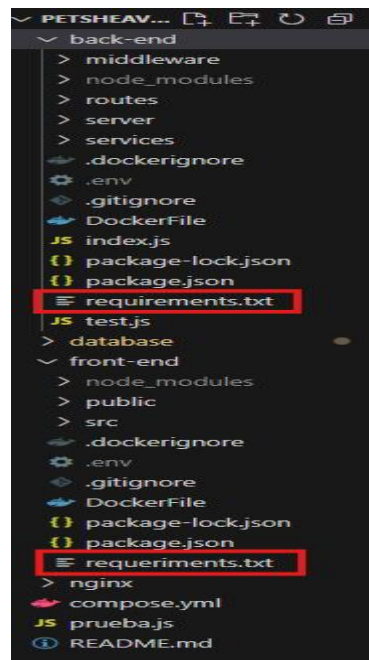
Manual Técnico



```
.env
back-end > .env
1 JWT_SECRET=pets_heaven_vite
2 PORT=3000
3 PORT_DB=3306
4 USER_DB=root
5 HOST_DB=localhost
6 NAME_DB=pets_heaven
7 PASSWORD_DB=""
```

Img 41- Datos en el archivo '.env'. Autoría propia

Paso 10: Con el fin de que el proyecto funcione e inicie sin problemas hay algunas dependencias que se deben descargar, cada sección (front y back) tienen las propias en cada archivo 'requirements.txt' de su sección correspondiente:



Img 42- Señalización de cada archivo 'requirements'. Autoría propia

paso 10.1: usamos la terminal que ya tenemos para instalar cada dependencia, iniciemos con las dependencias del back-end, para esto entramos al archivo 'requirements.txt' del back y copiamos línea por línea las



Manual Técnico

dependencias, en este caso solo hay una línea, y puede elegir entre instalarla con npm o con yarn si lo tiene instalado, en esta ocasión se usará npm:

```
requirements.txt X requirements.txt X
back-end > requirements.txt
1 cors == 2.8.5
2 express-rate-limit == 7.5.0
3 express == 5.1.0
4 mysql == 2.18.1
5 jsonwebtoken == 9.0.2
6 bcrypt == 5.1.1
7 dotenv == 16.0.3
8 cookie-parser == 1.4.6
9
10 ✨
11 npm i cors express mysql express-rate-limit jsonwebtoken bcrypt dotenv cookie-parser
12
13 yarn add cors express mysql express-rate-limit jsonwebtoken bcrypt dotenv cookie-parser
```

Img 43- Dependencias a usar. Autoría propia

una vez copiado se pegará en la terminal, pero antes debemos entrar a la carpeta back-end para esto en la terminal usaremos `cd back-end` y para luego pegar las dependencias para ser instaladas:

```
PS C:\Users\Aprendiz\Desktop\pets\PetsHeaven> cd .\back-end
PS C:\Users\Aprendiz\Desktop\pets\PetsHeaven\back-end> npm i cors express mysql express-rate-limit jsonwebtoken bcrypt dotenv cookie-parser
```

Img 44- Ejemplo de instalación de dependencias. Autoría propia

y se hace el mismo proceso en el front, pero primero debemos salir de la carpeta back-end, para esto se usa el comando `'cd ..'` que nos deja en este caso en el punto inicial, donde podemos usar de nuevo el comando `'cd front-end'` y copiamos y pegamos las dependencias del archivo `'requirements.txt'` del front:



Manual Técnico

```
requirements.txt x requirements.txt
front-end > requirements.txt
1 # Dep
2 react
3 react-dom
4 react-hook-form
5 boxicons
6 react-router
7 react-router-dom
8 lucide-react
9 @supabase/supabase-js
10 @emailjs/browser
11 sweetalert
12 sweetalert2
13 framer-motion
14 @fullcalendar/core
15 @fullcalendar/react
16 @fullcalendar/daygrid
17 @fullcalendar/interaction
18 @fullcalendar/timegrid
19 @fullcalendar/list
20 axios
21 sass-embedded
22 react-scripts
23
24 npm i react react-dom react-hook-form boxicons react-router react-router-dom lucide-react @supabase/supabase-js @emailjs/browser sweetalert sweetalert2 framer-motion
25 npm i @fullcalendar/core @fullcalendar/react @fullcalendar/daygrid @fullcalendar/interaction @fullcalendar/timegrid @fullcalendar/list axios compresorjs
26 npm i -D sass-embedded react-scripts
27
28 yarn add react react-dom react-hook-form boxicons react-router react-router-dom lucide-react @supabase/supabase-js @emailjs/browser sweetalert sweetalert2 framer-motion
29 yarn add @fullcalendar/core @fullcalendar/react @fullcalendar/daygrid @fullcalendar/interaction @fullcalendar/timegrid @fullcalendar/list axios compresorjs
30 yarn add -D sass-embedded react-scripts
```

Img 45- Dependencias a usar. Autoría propia

Esta vez son 3 líneas por lo cual debemos instalar una por una para evitar problemas:

```
PS C:\Users\Aprendiz\Desktop\pets\PetsHeaven\back-end> cd ..
PS C:\Users\Aprendiz\Desktop\pets\PetsHeaven> cd .\front-end
PS C:\Users\Aprendiz\Desktop\pets\PetsHeaven\front-end> npm i react react-dom react-hook-form boxicons react-router react-router-dom lucide-react @supabase/supabase-js @emailjs/browser sweetalert sweetalert2 framer-motion
```

Img 46- Señalización de la conexión a usar. Autoría propia

NOTA: EN LA IMAGEN SALE SOLO UNA LINEA DE DEPENDENCIAS SIENDO INSTALADAS, ESTA ES UN EJEMPLO DE COMO FUNCIONAN TODAS, POR LO CUAL CADA LINEA DEBE SER INSTALADA.

Paso 11: Ahora solo queda correr el proyecto, para esto se abren 2 terminales mas una para el back-end y la otra para el Front-end, para hacer esto debemos pulsar el botón mas (+) en la parte superior derecha de la terminal que ya abrimos:



Img 47- Creación mas terminales. Autoría propia



Manual Técnico

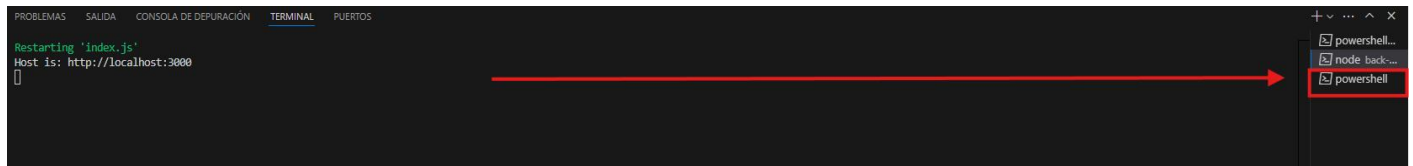
NOTA: PULSAMOS 2 VECES EL BOTON PARA CREAR LAS 2 TERMINALES QUE NECESITAMOS.

11.1: Para que el proyecto funcione se debe correr ambas partes back y front, iniciando con el back para esto usamos los siguientes comandos. 'cd back-end' (para entrar a la carpeta back-end) y luego el comando 'npm run dev' para iniciarlo, si todo salió bien debe salir de la siguiente manera:



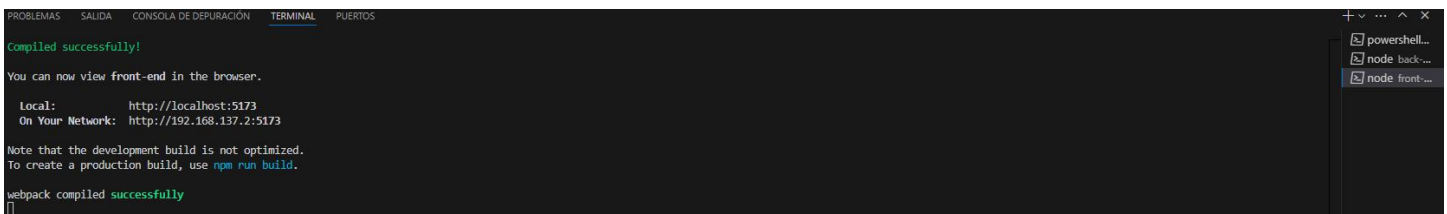
Img 48- Confirmación de uso del back-end. Autoría propia

Ahora solo debemos correr el front-end, para esto se creo anteriormente otra terminal, debemos ingresar a esa terminal que se sitúa donde se ve a continuación:



Img 49- Señalización de uso de otra terminal. Autoría propia

y hacemos el mismo procesamiento del back usamos el comando 'cd front-end' seguido del comando



'npm run dev' y si todo se realizó bien la terminal se debería ver así:

Img 50- Confirmación de uso del front-end. Autoría propia

Y seguidamente abrir el proyecto en su navegador predeterminado:



Centro Industrial y Desarrollo
Empresarial Soacha
Regional Cundinamarca

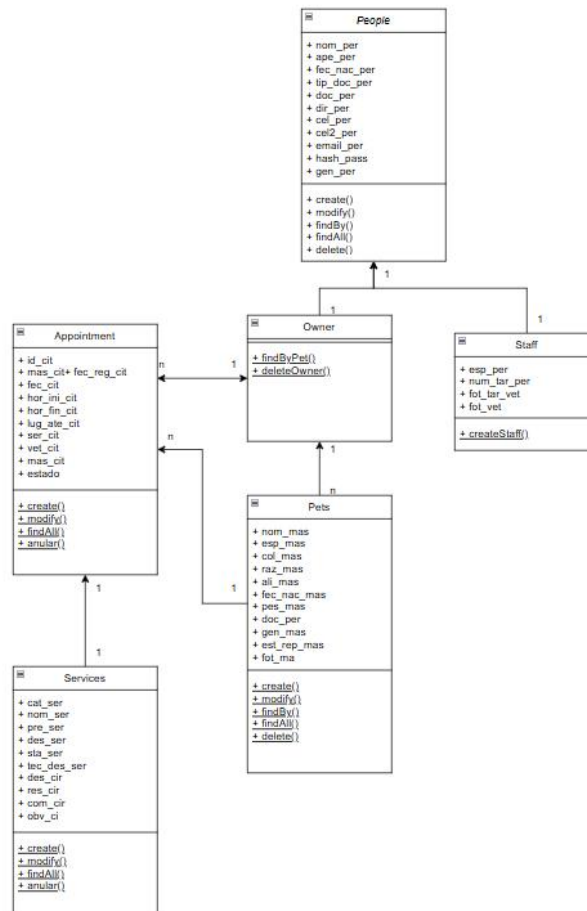


Manual Técnico



Img 51- Pagina funcionando. Autoría propia

12. Diagrama de Clases



Img 52- Diagrama de clases. Autoría propia

A continuación se describirían las clases y la relaciones que se están manejando en el sistema:

Clase People: Esta es la clase base de la cual heredan las clases Owner y Staff y que consta de todos los usuarios del sistema, Cuenta con atributos como Su nombre, tipo y documento de identidad fecha de nacimiento, etc. Y con los métodos:

- + create()
- + modify()
- + findBy()
- + findAll()
- + delete()



Manual Técnico

Clase Owner: Esta clase representa a todo dueño de mascotas heredando datos de la clase People como su Nombre, identidad, etc. Contando con los métodos:

+ findByPet()
+ deleteOwner()

Clase staff: Esta clase representa todo el personal de la veterinaria heredando datos de la clase people como su nombre, identidad, etc. Y agregando campos específicos de el como su especialidad y contando con el método + createStaff()

Clase pets: Esta clase representa las mascotas registradas en el sistema contando con atributos únicos como el nombre del animal, su color, raz, peso, etc contando con los métodos:

+ create()
+ modify()
+ findBy()
+ findAll()
+ delete()

Clase services: Esta clase representa todos los servicios que la veterinaria tiene para ofrecer contando Con los atributos como el nombre del servicio, precio, descripcion, etc y con los métodos:

+ create()
+ modify()
+ findAll()
+ anular()

Clase Appointment: Esta clase representa todas las citas medicas del sistema contando con atributos como el id de la cita, fecha, estado, etc. Tambien contando con los métodos:

+ create()
+ modify()
+ findAll()
+ anular()

Relaciones:

Owner y **Staff** heredan de **People**

Owner puede tener muchas mascotas (**pets**)

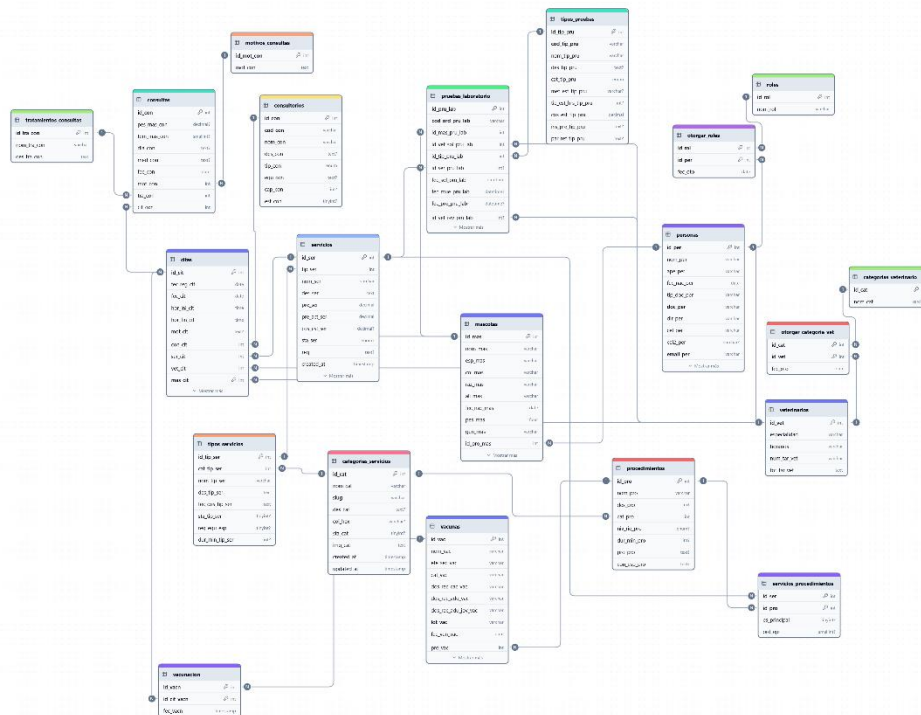
Appointment se conecta con:



Manual Técnico

- Una mascota (**Pets**)
- Un dueño (**Owner**)
- Un miembro del personal (**Staff**)
- Un servicio (**Services**)

13.Modelo entidad Relación



Img 53- Modelo de entidad relación. Autoría propia

A continuación, se explicará el diagrama de entidad relación.

El diagrama este compuesto por entidades (tablas) que representan cosas como:

- Personas
- Mascotas
- Consultas y procedimientos
- Citas, vacunas, tratamientos, etc.

Las cuales tienen relaciones entre si tales como:



Manual Técnico

-otorgar_roles

- id_rol → roles(id_rol)
- id_per → personas(id_per)

-veterinarios

- id_vet → personas(id_per)
(un veterinario es una persona)

-otorgar_categoria_vet

- id_cat → categorias_veterinario(id_cat)
- id_vet → veterinarios(id_vet)

-mascotas

- id_pro_mas → personas(id_per)
(el propietario de la mascota)

-tipos_servicios

- cat_tip_ser → categorias_servicios(id_cat)

-servicios

- tip_ser → tipos_servicios(id_tip_ser)

-procedimientos

- cat_pro → categorias_servicios(id_cat)

-servicios_procedimientos

- id_ser → servicios(id_ser)
- id_pro → procedimientos(id_pro)

-pruebas_laboratorio

- id_mas_pru_lab → mascotas(id_mas)
- id_vet_sol_pru_lab → veterinarios(id_vet)
- id_tip_pru_lab → tipos_pruebas(id_tip_pru)
- id_ser_pru_lab → servicios(id_ser)
- id_vet_rev_pru_lab → veterinarios(id_vet)

-citas

- con_cit → consultorios(id_con)
- ser_cit → servicios(id_ser)
- vet_cit → veterinarios(id_vet)
- mas_cit → mascotas(id_mas)

-vacunas

- pro_vac → procedimientos(id_pro)

-vacunacion

- id_vacn → vacunas(id_vac)
- id_cit_vacn → citas(id_cit)

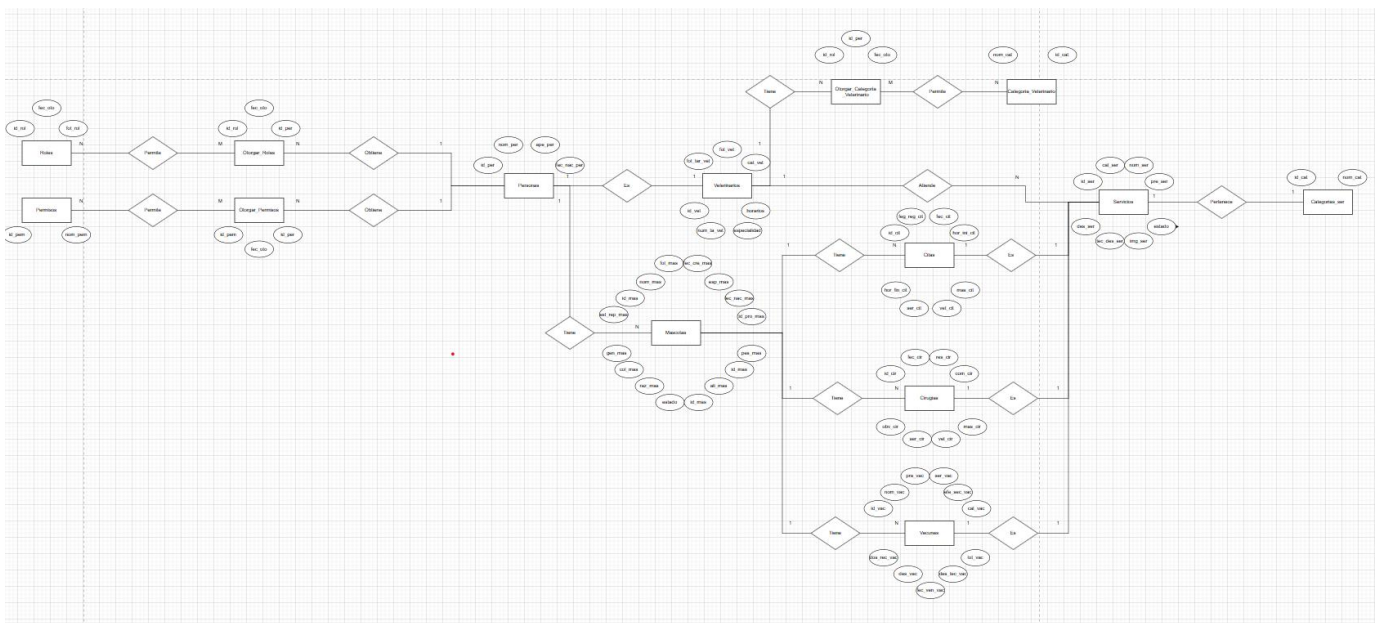
-consultas

- mot_con → motivos_consultas(id_mot_con)
- tra_con → tratamientos_consultas(id tra con)



- cit_con → citas(id_cit)

14. Diagrama entidad relación



Img 54- Diagrama entidad relación. Autoría propia

El siguiente diagrama representa el modelo entidad-relación de un sistema veterinario.

Su objetivo es organizar la información relacionada con personas, roles, mascotas, veterinarios, citas, servicios, cirugías, vacunas, categorías y permisos. Este modelo permite entender cómo se relacionan los diferentes actores y procesos en la gestión de una clínica veterinaria.



Manual Técnico

-Entidad: Personas

Contiene la información básica de cada individuo que interactúa con el sistema (clientes, veterinarios, personal, etc.).

-Roles y Permisos

El sistema tiene control de acceso mediante **roles** y **permisos**.

Una persona puede tener **uno o varios roles** (Otorga_Roles)

Un rol puede tener **muchos permisos** (Permite)

Los permisos también pueden ser otorgados directamente a una persona (Otorga_Permisos)

-Mascotas

Cada persona puede tener una o varias mascotas (Tiene).

-Veterinarios

Entidad independiente con especialización (especialidad), que **atienden citas** y pueden estar categorizados.

-Citas

Representa las consultas agendadas.

-Servicios

Servicios que ofrece la clínica.

-Vacunas

Cada vacuna tiene una categoría y puede aplicarse a una mascota.

-Cirugías

Registro de procedimientos quirúrgicos.

Relaciones clave



Manual Técnico

Personas puede tener muchos **roles** y **permisos** (relaciones N:M)

Mascotas se vinculan con personas, citas, vacunas y cirugías

Veterinarios pueden pertenecer a categorías y atender muchas citas

Servicios se agrupan por categorías

Todo está debidamente normalizado para evitar redundancia

15. Diccionario de Datos

A continuación, se mostrará el diccionario de datos del sistema:

Columna	Tipo de dato	Longitud máxima	¿Es nulo?	Valor por defecto	Comentario
id_cat	int	NULL	NO	NULL	ID de la categoría de servicio
nom_cat	varchar	100	NO	NULL	Nombre de la categoría
slug	varchar	100	NO	NULL	Versión normalizada y simplificada del nombre
des_cat	text	65535	YES	NULL	Descripción de la categoría
col_hex	varchar	7	YES	'#4b8ef5'	Color de la categoría
sta_cat	tinyint	NULL	YES		1 Estado activo/inactivo
img_cat	text	65535	NO	'No-Registrado'	Imagen de la categoría
created_at	timestamp	NULL	NO	current_timestamp()	Fecha de creación
updated_at	timestamp	NULL	NO	current_timestamp()	Fecha de última actualización
id_cat	int	NULL	NO	NULL	ID de la categoría
nom_cat	varchar	100	NO	NULL	Nombre de la categoría
id_cit	int	NULL	NO	NULL	ID de la cita
fec_reg_cit	date	NULL	NO	curdate()	Fecha de registro de la cita
fec_cit	date	NULL	NO	NULL	Fecha de la cita
hor_ini_cit	time	NULL	NO	NULL	Hora de inicio de la cita
hor_fin_cit	time	NULL	NO	NULL	Hora de fin de la cita
mot_cit	text	65535	YES	'No-registrado'	Motivo de la consulta
est_cit	enum	11	NO	NULL	Estado de la cita
fec_cre_cit	timestamp	NULL	NO	current_timestamp()	Fecha de creación del registro
fec_act_cit	timestamp	NULL	NO	current_timestamp()	Fecha de última actualización
con_cit	int	NULL	NO	NULL	ID del consultorio
ser_cit	int	NULL	NO	NULL	ID del servicio

Img 55. Autoría propia



Manual Técnico

vet_cit	int	NULL	NO	NULL	ID del veterinario
mas_cit	int	NULL	NO	NULL	ID de la mascota
id_con	int	NULL	NO	NULL	ID de la consulta
pes_mas_con	decimal	NULL	YES	NULL	Peso en kg
tem_mas_con	smallint	NULL	YES	NULL	Temperatura en °C
dia_con	text		65535 YES	NULL	Diagnóstico
med_con	text		65535 YES	NULL	Medicamentos recetados
fec_con	date	NULL	NO	curdate()	Fecha de la consulta
mot_con	int	NULL	NO	NULL	ID del motivo de consulta
tra_con	int	NULL	NO	NULL	ID del tratamiento aplicado
cit_con	int	NULL	NO	NULL	ID de la cita asociada
id_con	int	NULL	NO	NULL	ID único del consultorio
cod_con	varchar		10 NO	NULL	Código identificador del consultorio
nom_con	varchar		100 NO	NULL	Nombre del consultorio
des_con	text		65535 YES	'No-registrado'	Descripción del consultorio
tip_con	enum		15 NO	NULL	Tipo de consultorio
equ_con	text		65535 YES	'No-registrado'	Equipamiento disponible en formato JSON
cap_con	int	NULL	YES		1 Capacidad máxima del consultorio
est_con	tinyint	NULL	YES		1 Estado de actividad del consultorio
id_mas	int	NULL	NO	NULL	ID de la mascota
nom_mas	varchar		100 NO	NULL	Nombre de la mascota
esp_mas	varchar		100 NO	NULL	Especie de la mascota
col_mas	varchar		100 NO	NULL	Color de la mascota
raz_mas	varchar		100 NO	NULL	Raza de la mascota

Img 56. Autoría propia

mascotas	ali_mas	varchar		100 NO	NULL	Alimentación de la mascota
mascotas	fec_nac_mas	date	NULL	NO	NULL	Fecha de nacimiento de la mascota
mascotas	pes_mas	float	NULL	NO	NULL	Peso de la mascota
mascotas	gen_mas	varchar		20 NO	NULL	Género de la mascota
mascotas	id_pro_mas	int	NULL	NO	NULL	ID del propietario
mascotas	est_rep_mas	varchar		100 NO	NULL	Estado reproductivo
mascotas	estado	tinyint	NULL	NO		1 Estado activo/inactivo
mascotas	fot_mas	text		65535 NO	NULL	Foto de la mascota
mascotas	fec_cre_mas	date	NULL	NO	curdate()	Fecha de creación del registro
motivos_consultas	id_mot_con	int	NULL	NO	NULL	ID del motivo de consulta
motivos_consultas	mot_con	text		65535 NO	NULL	Motivo de consulta
otorgar_categoria_vet	id_cat	int	NULL	NO	NULL	ID de la categoría
otorgar_categoria_vet	id_vet	int	NULL	NO	NULL	ID del veterinario
otorgar_categoria_vet	fec_oto	date	NULL	NO	curdate()	Fecha de otorgamiento
otorgar_rol	id_rol	int	NULL	NO	NULL	ID del rol otorgado
otorgar_rol	id_per	int	NULL	NO	NULL	ID de la persona
otorgar_rol	fec_oto	date	NULL	NO	curdate()	Fecha de otorgamiento
personas	id_per	int	NULL	NO	NULL	ID de la persona
personas	nom_per	varchar		100 NO	NULL	Nombre de la persona
personas	ape_per	varchar		100 NO	NULL	Apellido de la persona
personas	fec_nac_per	date	NULL	NO	NULL	Fecha de nacimiento
personas	tip_doc_per	varchar		10 NO	NULL	Tipo de documento
personas	doc_per	varchar		20 NO	NULL	Número de documento
personas	dir_per	varchar		100 NO	NULL	Dirección

Img 57. Autoría propia



Manual Técnico

personas	cel_per	varchar		20 NO	NULL	Celular principal
personas	cel2_per	varchar		20 YES	NULL	Celular secundario
personas	email_per	varchar		100 NO	NULL	Correo electrónico
personas	cont_per	varchar		255 NO	NULL	Contraseña
personas	gen_per	varchar		100 NO	NULL	Género
personas	estado	tinyint	NULL	NO		1 Estado activo/inactivo
personas	fot_per	text		65535 NO	'https://img.freepik.com/'	Foto de perfil
personas	fec_cre_per	date	NULL	NO	curdate()	Fecha de creación del registro
procedimientos	id_pro	int	NULL	NO	NULL	ID del procedimiento
procedimientos	nom_pro	varchar		100 NO	NULL	Nombre del procedimiento
procedimientos	des_pro	text		65535 NO	NULL	Descripción del procedimiento
procedimientos	cat_pro	int	NULL	NO	NULL	ID de la categoría del procedimiento
procedimientos	niv_rie_pro	enum		8 YES	NULL	Nivel de riesgo del procedimiento
procedimientos	dur_min_pro	int	NULL	YES	NULL	Duración mínima del procedimiento en minutos
procedimientos	pro_pro	text		65535 YES	'No-registrado'	Protocolo / Pasos del procedimiento
procedimientos	con_esp_pro	text		65535 YES	'No-registrado'	Consideraciones especiales
pruebas_laboratorio	id_pru_lab	int	NULL	NO	NULL	ID único de la prueba de laboratorio
pruebas_laboratorio	cod_ord_pru_lab	varchar		20 NO	NULL	Código de orden de la prueba
pruebas_laboratorio	id_mas_pru_lab	int	NULL	NO	NULL	ID de la mascota
pruebas_laboratorio	id_vet_sol_pru_lab	int	NULL	NO	NULL	ID del veterinario solicitante
pruebas_laboratorio	id_tip_pru_lab	int	NULL	NO	NULL	ID del tipo de prueba
pruebas_laboratorio	id_ser_pru_lab	int	NULL	YES	NULL	ID del servicio asociado (opcional)
pruebas_laboratorio	fec_sol_pru_lab	datetime	NULL	NO	NULL	Fecha y hora de solicitud de la prueba
pruebas_laboratorio	fec_mue_pru_lab	datetime	NULL	YES	NULL	Fecha y hora de toma de muestra

Img 58. Autoría propia

pruebas_laboratorio	fec_pro_pru_lab	datetime	NULL	YES	NULL	Fecha y hora de procesamiento de la muestra
pruebas_laboratorio	fec_res_pru_lab	datetime	NULL	YES	NULL	Fecha y hora de resultado disponible
pruebas_laboratorio	est_pru_lab	enum		14 YES	'REGISTRADO'	Estado de la prueba
pruebas_laboratorio	pri_pru_lab	enum		7 YES	'ROUTINA'	Prioridad de la prueba
pruebas_laboratorio	obs_mue_pru_lab	text		65535 YES	NULL	Observaciones sobre la muestra
pruebas_laboratorio	cos_fin_pru_lab	decimal	NULL	YES	0.00	Costo final de la prueba
pruebas_laboratorio	res_pru_lab	text		65535 YES	'No-registrado'	Resultados de la prueba en formato
pruebas_laboratorio	id_vet_rev_pru_lab	int	NULL	YES	NULL	ID del veterinario revisor
roles	id_rol	int	NULL	NO	NULL	ID del rol
roles	nom_rol	varchar		100 NO	NULL	Nombre del rol
servicios	id_ser	int	NULL	NO	NULL	ID del servicio
servicios	tip_ser	int	NULL	NO	NULL	ID del tipo de servicio
servicios	nom_ser	varchar		100 NO	NULL	Nombre del servicio
servicios	des_ser	text		65535 NO	NULL	Descripción del servicio
servicios	pre_ser	decimal	NULL	NO	NULL	Precio base del servicio
servicios	pre_act_ser	decimal	NULL	NO	NULL	Precio actual del servicio
servicios	cos_est_ser	decimal	NULL	YES	NULL	Costo estimado del servicio
servicios	sta_ser	enum		13 YES	'DISPONIBLE'	Estado del servicio
servicios	req	text		65535 YES	'No-registrado'	Requisitos previos para el servicio
servicios	created_at	timestamp	NULL	NO	current_timestamp()	Fecha de creación
servicios	updated_at	timestamp	NULL	NO	current_timestamp()	Fecha de última actualización
servicios_procedimientos	id_ser	int	NULL	NO	NULL	ID del servicio
servicios_procedimientos	id_pro	int	NULL	NO	NULL	ID del procedimiento
servicios_procedimientos	es_principal	tinyint	NULL	YES		1 Indica si el procedimiento es principal

Img 59. Autoría propia



Manual Técnico

servicios_procedimientos	ord_eje	smallint	NULL	YES	NULL	Orden de ejecución del procedimiento
tipos_pruebas	id_tip_pru	int	NULL	NO	NULL	ID único del tipo de prueba
tipos_pruebas	cod_tip_pru	varchar		20 NO	NULL	Código identificador de la prueba
tipos_pruebas	nom_tip_pru	varchar		100 NO	NULL	Nombre del tipo de prueba
tipos_pruebas	des_tip_pru	text		65535 YES	'No-registrado'	Descripción de la prueba
tipos_pruebas	cat_tip_pru	enum		13 NO	NULL	Categoría de la prueba
tipos_pruebas	met_est_tip_pru	varchar		100 YES	NULL	Método estándar utilizado
tipos_pruebas	tie_est_hrs_tip_pru	int	NULL	YES	NULL	Tiempo estimado en horas para el resultado
tipos_pruebas	cos_est_tip_pru	decimal	NULL	NO	NULL	Costo estimado en el resultado
tipos_pruebas	ins_pre_tip_pru	text		65535 YES	'No-registrado'	Instrucciones de preparación para la prueba
tipos_pruebas	par_ref_tip_pru	text		65535 YES	'No-registrado'	Parámetros de referencia
tipos_servicios	id_tip_ser	int	NULL	NO	NULL	ID del tipo de servicio
tipos_servicios	cat_tip_ser	int	NULL	NO	NULL	ID de la categoría del tipo de servicio
tipos_servicios	nom_tip_ser	varchar		100 NO	NULL	Nombre del tipo de servicio
tipos_servicios	des_tip_ser	text		65535 NO	NULL	Descripción del tipo de servicio
tipos_servicios	tec_des_tip_ser	text		65535 NO	NULL	Descripción técnica del tipo de servicio
tipos_servicios	sta_tip_ser	tinyint	NULL	YES		1 Estado del tipo de servicio
tipos_servicios	req_equ_esp	tinyint	NULL	YES		0 Define si requiere equipo especial
tipos_servicios	dur_min_tip_ser	int	NULL	YES	NULL	Duración mínima del servicio en horas
tratamientos_consultas	id_tra_con	int	NULL	NO	NULL	ID del tratamiento aplicado
tratamientos_consultas	nom_tra_con	varchar		100 NO	NULL	Tratamiento aplicado
tratamientos_consultas	des_tra_con	text		65535 NO	NULL	Descripción del tratamiento
vacunacion	id_vacn	int	NULL	NO	NULL	ID de la vacuna
vacunacion	id_cit_vacn	int	NULL	NO	NULL	ID de la cita

Img 60. Autoría propia

vacunacion	fec_vacn	timestamp	NULL	NO	current_timestamp()	Fecha de vacunación
vacunas	id_vac	int	NULL	NO	NULL	ID de la vacuna
vacunas	nom_vac	varchar		255 NO	NULL	Nombre de la vacuna
vacunas	efe_sec_vac	varchar		255 NO	NULL	Efectos secundarios de la vacuna
vacunas	cat_vac	varchar		100 NO	NULL	Categoría de la vacuna
vacunas	dos_rec_cac_vac	varchar		100 NO	NULL	Dosis recomendada cachorro
vacunas	dos_rec_adu_vac	varchar		100 NO	NULL	Dosis recomendada adulto
vacunas	dos_rec_adu_jov_vac	varchar		100 NO	NULL	Dosis recomendada adulto joven
vacunas	lot_vac	varchar		255 NO	NULL	Lote de la vacuna
vacunas	fec_ven_vac	date	NULL	NO	NULL	Fecha de vencimiento
vacunas	fec_cre_vac	date	NULL	NO	NULL	Fecha de creación
vacunas	fre_vac	int	NULL	NO	NULL	Frecuencia de vacunación en días
vacunas	des_vac	text		65535 YES	'No-registrado'	Descripción de la vacuna
vacunas	pre_vac	decimal	NULL	NO	NULL	Precio de la vacuna
vacunas	sta_vac	tinyint	NULL	NO		1 Estado de la vacuna (disponible / no disponible)
vacunas	pro_vac	int	NULL	NO	NULL	ID del procedimiento asociado
vacunas	created_at	timestamp	NULL	NO	current_timestamp()	Fecha de creación
vacunas	updated_at	timestamp	NULL	NO	current_timestamp()	Fecha de última actualización
veterinarios	id_vet	int	NULL	NO	NULL	ID del veterinario (persona)
veterinarios	especialidad	varchar		100 NO	NULL	Especialidad del veterinario
veterinarios	horarios	varchar		100 NO	NULL	Horarios de atención
veterinarios	num_tar_vet	varchar		100 NO	'no-registrado'	Número de tarjeta profesional
veterinarios	fot_tar_vet	text		65535 NO	'no-registrado'	Foto de la tarjeta profesional

Img 61-Diccionario de datos. Autoría propia

16.Glosario

-Aplicación

Sistema web desarrollado para que los usuarios interactúen con los servicios ofrecidos. En este caso, permite agendar citas, registrar tratamientos, gestionar vacunas, entre otros.

-Arquitectura

Estructura lógica y técnica del sistema. Define cómo se organiza la aplicación en componentes como frontend, backend y base de datos. Puede seguir modelos como MVC o arquitectura en capas.

-Datos

Información gestionada por el sistema. Incluye los registros de mascotas, propietarios, consultas, vacunas, etc. Su correcta gestión implica validación, almacenamiento y protección.



Manual Técnico

-Despliegue

Proceso de publicar la aplicación en internet. Involucra subir archivos a un servidor, configurar servicios en la nube y permitir el acceso público o restringido según el caso.

-Gestión

Módulo o conjunto de funcionalidades que permiten administrar recursos del sistema como usuarios, servicios, turnos, citas y stock médico.

-Objetivos

Metas específicas que busca alcanzar el desarrollo del sistema. Por ejemplo: facilitar la programación de citas, mejorar el seguimiento clínico de mascotas o automatizar el control de vacunas.

-Relaciones

Conexiones entre entidades del sistema. Pueden ser entre tablas de la base de datos (ej. mascota → persona) o entre componentes (ej. frontend → backend por medio de APIs).

-Requisitos

Condiciones que debe cumplir el sistema para funcionar correctamente. Se dividen en:

- Funcionales: qué debe hacer (agendar, consultar, editar).
- No funcionales: cómo debe hacerlo (rápido, seguro, escalable).

Resultados

Información generada después de ejecutar acciones en el sistema. Ejemplos: confirmación de cita, informe clínico, historial de atención.

-Servidor

Equipo físico o virtual donde se ejecuta la lógica de la aplicación (backend) y se almacena la base de datos. Recibe solicitudes del usuario y responde con datos o contenido.

-Veterinaria

Dominio específico del sistema. Engloba todas las funcionalidades necesarias para administrar una clínica veterinaria: pacientes (mascotas), médicos (veterinarios), servicios, historia clínica, etc.

-Web

Plataforma de acceso para los usuarios. La aplicación se utiliza mediante navegadores (Chrome, Firefox, etc.), con tecnologías como HTML, CSS, JavaScript y frameworks como React o Vue.



17.Referencias

Visual Studio Code

<https://code.visualstudio.com/>

https://en.wikipedia.org/wiki/Visual_Studio_Code

XAMPP

<https://draftdesignweb.com/2023/12/25/xampp-una-solucion-integral-para-desarrolladores/>

<https://www.apachefriends.org/es/index.html>

Supabase

<https://www.0hands.com/tools/supabase>

<https://supabase.com/>

Docker

<https://www.yoelprogramador.com/tutorial-de-docker/>

<https://www.docker.com/>

Express.js

<https://ajeetchaulagain.com/blog/express-js-getting-started>

<https://expressjs.com/>

Node.js

<https://www.edureka.co/blog/learn-node-js/>

<https://nodejs.org/en>

CORS

<https://www.agilicus.com/cors-and-you/>



Manual Técnico

MySQL

<https://aws.amazon.com/es/rds/mysql/>

<https://www.mysql.com/>

JWT (JSON Web Token)

<https://worldvectorlogo.com/logo/jwtio-json-web-token>

<https://jwt.io/>

Bcrypt

<https://github.com/topics/bcrypt-library>

<https://www.npmjs.com/package/bcrypt>

Dotenv

<https://webcatalog.io/es/apps/dotenv>

<https://www.npmjs.com/package/dotenv>

FullCalendar

<https://community.weweb.io/t/full-calendar-npm-plugin/9854>

<https://fullcalendar.io/>

EmailJS

<https://www.emailjs.com/>

React DOM

<https://www.npmjs.com/package/react-dom>

React Hook Form

<https://react-hook-form.com/>



Centro Industrial y Desarrollo
Empresarial Soacha
Regional Cundinamarca



Manual Técnico

React Router

<https://reactrouter.com/en/main>

React general

<https://es.wikipedia.org/wiki/React>

<https://es.react.dev/>

SweetAlert

<https://www.drupal.org/project/sweetalert>

<https://sweetalert2.github.io/>

Axios

<https://www.axios.com/press-past-releases/cox-enterprises-acquires-axios-media-inc>

<https://axios-http.com/es/docs/intro>

CSS

<https://www.jasoft.org/Blog/post/como-usar-los-pseudo-elementos-css-before-y-after-con-imagenes-en-html>