SOFT VET

SISTEMA DE GESTION PARA VETERINARIAS

Manual Técnico

Desarrolladores:

Ángel Pastrana

Joaquín Diaz

Mariano Celiz

David Valdez Gramajo

Información para mantenimiento y evolución del sistema

Este sistema está diseñado para permitir una fácil actualización y ampliación de funcionalidades.
El documento describe la estructura tecnológica utilizada, los componentes principales, los puntos de integración y los procedimientos necesarios para mantener el proyecto operativo y actualizado.
Incluye lineamientos para:

- Modificación y ampliación de módulos.
- Corrección de errores.
- Ajustes de seguridad.
- Optimización de rendimiento.
- Actualización de dependencias.

Arquitectura, configuración y procedimientos de actualización

Arquitectura general

Frontend**:** desarrollado en React, empleando componentes funcionales, Hooks, React Router para navegación y Axios para comunicación con la API.

Backend**:** desarrollado con Node.js y express.js Gestiona autenticación, validaciones y la lógica de negocio.

API REST**:** comunicación entre frontend y backend mediante peticiones HTTP con respuestas en formato JSON.

Base de Datos**:** MYSQL utilizada para almacenar todos los registros del sistema.

Estructura Cliente–Servidor: separación completa entre interfaz y servicios.

Configuracion del entorno de desarrollo

1- Clonar repositoio
2- Instalar dependencias del fronten en /frontend:
    a. npm install
3- Configurar archivo .env con
    a. Variables de entornos necesarias
4- Iniciar el proyecto en modo desarrollo
    a. npm run dev
5- Ejecutar backend en modo desarrollo
    a. Node –watch index.js

Procedimientos de actualización

1- Actualizar repositorio local con los cambios recientes.
2- Crear rama para el desarrollo de nuevas funciones o correcciones.
3- Realizar modificaciones necesarias en frontend, backend o base de datos.

4- Ejecutar pruebas locales y validar funcionamiento.
5- Generar el build(en caso de ser necesario) del frontend:
    a. Npm run build
6- Subir el build al servidor o servicio de hosting.
7- Actualizar el backend y aplicar migraciones de base de datos si corresponden.
8- Verificar funcionamiento general del sistema después de la actualización.

Secciones minimas

Requisitos del sistema:

Node.js: versión recomendada (18.x o superior)

React: versión estable incluida en el proyecto.

Dependencias principales:

Frontend:

```
    "axios": "^1.12.2",
    "bootstrap": "^5.3.8",
    "bootstrap-icons": "^1.13.1",
    "lucide-react": "^0.552.0",
    "react": "^19.1.1",
    "react-bootstrap": "^2.10.10",
    "react-bootstrap-icons": "^1.11.6",
    "react-dom": "^19.1.1",
    "react-icons": "^5.5.0",
    "react-router-dom": "^7.9.4",
    "sweetalert2": "^11.26.3",
    "zustand": "^5.0.8"
```

Backend:

```
    "bcrypt": "^6.0.0",
    "cookie-parser": "^1.4.7",
    "cors": "^2.8.5",
    "dotenv": "^17.2.3",
    "express": "^5.1.0",
    "jsonwebtoken": "^9.0.2",
    "morgan": "^1.10.1",
    "mysql2": "^3.15.1"
```

Diseño del sistema:

Estructura principal del Frontend:

/src
- /assets
- /components
- /endpoints
- /hooks
- /pages
- /routers
- /validations
- /zustand

/src
Directorio raíz donde se encuentra todo el código fuente del frontend.

/assets
Contiene imágenes, íconos, estilos y otros recursos estáticos utilizados en la interfaz del sistema.

/components
Incluye los componentes reutilizables de interfaz, como formularios, tablas, botones, modales y elementos visuales que se utilizan en distintas partes del sistema.

/endpoints
Carpeta donde se centralizan todas las URLs o funciones relacionadas a los endpoints de la API, permitiendo una gestión organizada de las rutas de comunicación con el backend.

/hooks
Contiene hooks personalizados utilizados para encapsular lógica reutilizable (manejo de formularios, consultas, validaciones, etc.).

/pages
Agrupa las pantallas principales del sistema, como Empleados, Mascotas, Clientes, Turnos, Ventas, Dashboard, entre otras.

/routers
Define la configuración de navegación del sistema, gestionando las rutas que conectan cada página y controlando el acceso según permisos.

/validations
Incluye funciones, esquemas o utilidades para validar los datos ingresados en formularios o procesos específicos del sistema.

/zustand
Carpeta destinada al manejo del estado global del sistema utilizando **Zustand**, donde se crean y administran los stores que controlan datos compartidos entre componentes (auth, usuario, configuraciones, etc.).

Estructura principal del backend:

/config

/controllers

/middlewares

/routers

/service

/validations

/config
Contiene los archivos de configuración del sistema, como conexión a la base de datos, variables de entorno, configuración de seguridad y parámetros generales utilizados por el servidor.

/controllers
Incluye los controladores responsables de gestionar la lógica de cada módulo del sistema. Aquí se procesan las solicitudes recibidas desde los routers, se coordinan llamadas a servicios y se generan las respuestas para la API.

/middlewares
Agrupa funciones que se ejecutan antes o durante el manejo de una petición, como validaciones de autenticación, permisos, manejo de errores, logs, protección de rutas o filtrado de datos.

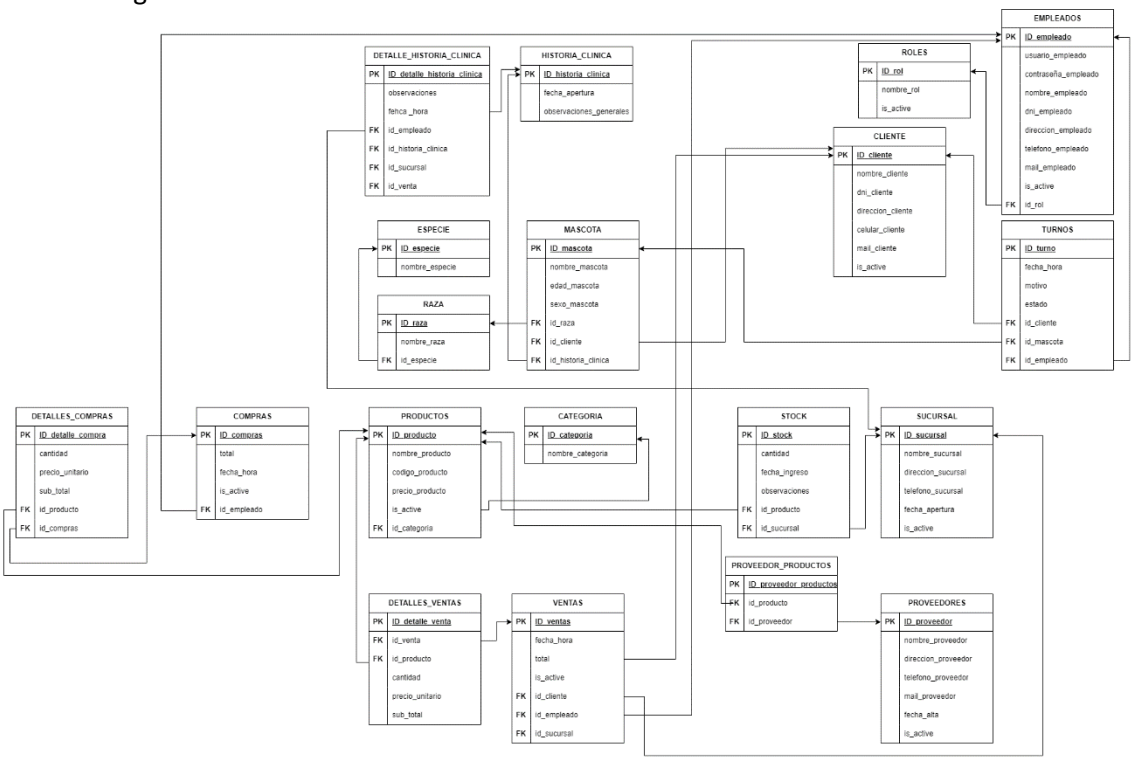
/routers
Define las rutas de la API, asociando cada endpoint con su respectivo controlador y middleware. Es la capa que organiza las URL del sistema y su comportamiento.

/service
Contiene la lógica de negocio y operaciones principales del sistema, como consultas a la base de datos, procesamiento de información o interacción con otros servicios. Los controladores se apoyan en esta capa para mantener el código limpio y organizado.

/validations
Incluye funciones o esquemas de validación utilizados para verificar datos entrantes (por ejemplo, en formularios o peticiones), garantizando que cumplan con el formato y los requisitos antes de ser procesados por el sistema.

Estructura de base de datos

Diagrama entidad relacion



Api:

La comunicación con el backend se realiza mediante endpoints REST.

Empleados		
GET	/empelados	Obtiene todos lo empleados
POST	/empleados/crear	Crea un nuevo empleado
PUT	/empleados/editar/:id	Edita un empleado
POST	/empleados/autenticar	Autentifica un empleado
POST	/empleado/info	Obtiene información de empleado
GET	/empleados/ver	Obtiene todos los empleados
GET	/empleados/ver/:id	Obtiene un empleado por id
GET	/empleados/logout	Cierra sesión de un empleado