**1. Resumen del Proyecto**

El sistema de gestión vehicular permite registrar, consultar, editar y eliminar vehículos, técnicos y mantenimientos realizados. Está desarrollado en Java, con interfaz gráfica usando Swing y persistencia en MySQL. Su objetivo es facilitar el control de mantenimientos asociados a vehículos de forma eficiente.

**2. Objetivos**

**Objetivo general:**

Desarrollar un sistema de escritorio que gestione la información de vehículos, técnicos y mantenimientos, con operaciones CRUD y base de datos relacional.

**Objetivos específicos:**

- Implementar un CRUD para vehículos.  
- Implementar un CRUD para técnicos.  
- Asociar mantenimientos a técnicos y vehículos.  
- Crear una interfaz gráfica sencilla e intuitiva.  
- Documentar el código y las decisiones técnicas.

**3. Tecnologías utilizadas**

|  |  |
| --- | --- |
| **Componente** | **Tecnología** |
| Lenguaje de programación | Java |
| Interfaz gráfica | Swing |
| Base de datos | MySQL |
| Conexión a BD | JDBC |
| IDE | NetBeans |
| Gestor de dependencias | Maven |

**4. Modelo de Base de Datos**

El modelo de base de datos incluye las siguientes tablas principales:  
- tb\_usuario: usuarios del sistema.  
- tb\_tecnico: técnicos registrados.  
- tb\_categoria: categorías de vehículos.  
- tb\_vehiculo: vehículos con sus características.  
- tb\_cabecera\_mantenimiento: mantenimientos realizados.  
- tb\_detalle\_mantenimiento: trabajos realizados por mantenimiento.

**5. Relación base de datos**

Diagrama

El contenido generado por IA puede ser incorrecto.

**6. Estructura del Proyecto**

/src  
 ├── conector/ → Contiene los archivos .jar utilizados en el proyecto  
 ├── conexión/ → conexión a la base de datos  
 ├── controlador/ → Lógica del negocio  
 ├── img/ → Recursos visuales para el proyecto

├── modelo/ → Contiene las clases del proyecto   
 └── vista/ → Formularios del proyecto

**7. Decisiones de diseño**

- Se usó JDBC por su simplicidad.  
- Se usaron PreparedStatement para evitar inyecciones SQL.  
- Se agregó campo 'estado' en tablas para eliminaciones lógicas.

- **Paleta de colores**

Para el diseño de la interfaz gráfica del sistema de gestión vehicular, se optó por una paleta de colores compuesta por tonos **azules**, **blancos** y **oscuros** (gris, negro o variantes). Esta elección se basó en criterios de **usabilidad, estética profesional y comodidad visual** para el usuario.

* **Azul:** transmite confianza, seguridad y profesionalismo, características deseables para un sistema administrativo o técnico.
* **Blanco:** aporta claridad, limpieza visual y facilita la legibilidad de los textos.
* **Colores oscuros:** se usaron como fondo o contraste para dar enfoque a los elementos principales, evitar fatiga visual y lograr un diseño moderno.

Esta combinación busca ofrecer una interfaz limpia, intuitiva y amigable para el usuario, especialmente en contextos donde se trabaja muchas horas frente a la pantalla

**8. Manual de instalación y ejecución**

1. Crear la base de datos con el script SQL.  
2. Configurar los datos de conexión en la clase Conexion.java.  
3. Ejecutar la clase Main.java o el formulario principal.  
4. Verificar que se conecte a la base de datos correctamente.

**9. Conclusiones**

- El sistema cumple con las funciones básicas.  
- Permite llevar un control claro de vehículos y mantenimientos.  
- Puede escalarse con más módulos como reportes, filtros, exportaciones, etc.

**10. Anexos**

Se incluye:  
- Código fuente del sistema.  
- Script SQL para la creación de la base de datos.