"SISTEMA DE GESTIÓN VEHICULAR - JAVA + MYSQL"

Catedra: Programación II

Integrantes del grupo:

Castro, Claudio David - Matricula 101132

Castro, Ramón Gabriel – Matricula 101136

Salas, Rocío Soledad – Matricula 101138

Torres, Jorge David- Matricula 101129

Año 2025

❖ Realización del trabajo

Castro Ramón Gabriel: Capa Models

Responsabilidades principales:

- Diseño e implementación de entidades (Base.java, Vehiculo.java, SeguroVehicular.java)
- Definición de relaciones entre clases
- Implementación de getters, setters, constructores y toString()
- Documentación de decisiones de diseño

Tareas específicas:

- Crear estructura de herencia en Models
- Definir atributos y métodos de cada entidad
- Asegurar coherencia entre modelo Java y base de datos
- Revisar integración entre capas

Castro Claudio David: Base de Datos y DAO

Responsabilidades principales:

- Diseño del esquema de base de datos
- Implementación de la capa DAO (GenericDAO.java, VehiculoDAO.java, SeguroVehicularDAO.java)
- Optimización de consultas SQL
- Configuración de conexión a BD (DatabaseConnection.java)

Tareas específicas:

- Crear scripts SQL (estructura y datos de prueba)
- Implementar operaciones CRUD en DAOs
- Manejar transacciones y conexiones
- Optimizar consultas con JOINs e índices

Torres Jorge David: Lógica de Negocio

Responsabilidades principales:

- Implementación de la capa Service (GenericService.java, VehiculoServiceImpl.java, SeguroServiceImpl.java)
- Validaciones de negocio
- Coordinación entre DAOs para operaciones complejas
- Manejo de excepciones de negocio

Tareas específicas:

- Implementar validaciones de datos
- Aplicar reglas de negocio
- Orquestar operaciones entre múltiples DAOs
- Documentar reglas de negocio

Salas Rocío Soledad: UI y Pruebas de funcionamiento

Responsabilidades principales:

- Desarrollo de la interfaz de usuario (AppMenu.java, Main.java)
- Pruebas integrales del sistema
- Preparación de demostraciones

Tareas específicas:

- Implementar menú interactivo
- Validar flujo de usuario
- Ejecutar pruebas de funcionalidad
- Capturar evidencias de funcionamiento
- Preparar video demostrativo

❖ Dominio Seleccionado: Vehiculo-SeguroVehicular

Se desarrolló un sistema de gestión vehicular para administrar vehículos y sus pólizas de seguro asociadas. El sistema permite realizar operaciones CRUD completas sobre dos entidades, con una relación opcional entre ellas a través de una FK, esta relación vehículo-seguro es opcional, lo que implica el manejo de foreign keys nullables y consultas con JOINs condicionales.

El sistema implementa las cuatro operaciones fundamentales de persistencia: creación, lectura, actualización y eliminación de registros. Cada operación presenta desafíos técnicos específicos, es decir se definieron reglas de negocio medibles y específicas como por ejemplo validación de formato de dominio, unicidad de identificadores, rangos numéricos para años de fabricación, y controles de integridad referencial. Estas reglas se implementaron principalmente en la capa Service.

El sistema sigue una arquitectura en capas bien definida: presentación (Main), lógica de negocio (Service), acceso a datos (DAO), y entidades de dominio (Models). Esta separación de responsabilidades permite un desarrollo más organizado y facilita las pruebas. Esta arquitectura en capas permite extensiones futuras como la incorporación de módulos para propietarios, historial de seguros, sistema de multas, o mantenimiento preventivo, demostrando la escalabilidad del diseño.

En este proyecto se implementó el patrón DAO (Data Access Object) para abstraer el acceso a datos, y el patrón Service para organizar la lógica de negocio, también integra Java con MySQL utilizando JDBC, aplicando mejores prácticas como el uso de PreparedStatement para prevenir inyección SQL, y try-with-resources para el manejo automático de recursos.

- ❖ Diseño del Sistema

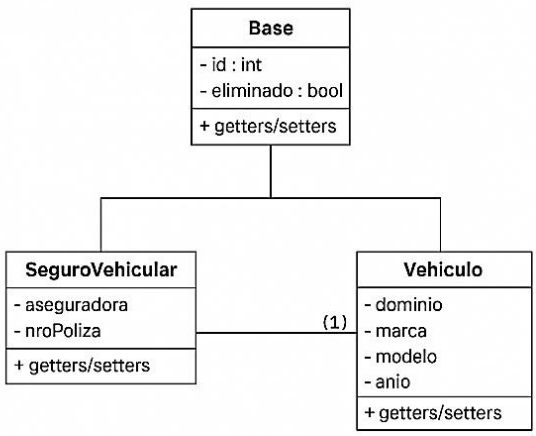
Con respecto al diseño del sistema se optó por una relación uno-a-uno opcional entre vehículos y seguros esta decisión permite que un vehículo exista sin seguro, manteniendo la flexibilidad del modelo. Se descartó la opción de primary key compartida por considerarse muy rígida para los requisitos del sistema.

Se implementó borrado lógico en lugar de físico, mediante un campo "eliminado" en ambas tablas. Esto preserva el historial completo de registros mientras mantiene la base de datos limpia para consultas habituales. Todas las consultas del sistema incluyen el filtro "WHERE eliminado = FALSE" para excluir registros inactivos.

El sistema se modeló con tres clases principales: la clase abstracta Base, y las clases Vehiculo y SeguroVehicular. La clase Base contiene los atributos comunes id y eliminado, de los cuales heredan las otras dos clases. Vehículo contiene atributos específicos como dominio, marca, modelo y anio, además de una referencia opcional a SeguroVehicular, este a su vez tiene como atributos propios a aseguradora y nroPoliza. La relación entre Vehiculo y SeguroVehicular es uno-a-uno opcional.

Con respecto a las Capas, la capa Models contiene las entidades del dominio, representando los objetos de negocio. La capa DAO se encarga del acceso a datos, abstrayendo las operaciones de base de datos. La capa Service contiene la lógica de negocio y validaciones. La capa Config gestiona la configuración de conexión a base de datos, y la capa Main coordina la interfaz de usuario y el flujo de la aplicación. Esta separación permite un mantenimiento sencillo y la independencia entre componentes.

Diagrama UML utilizado en el proyecto:



- Persistencia de Datos

Se diseñó una base de datos MySQL con dos tablas. La tabla "seguro_vehicular" almacena los datos de las pólizas de seguro, incluyendo id (clave primaria autoincremental), aseguradora (obligatoria), nroPoliza (único) y eliminado (para borrado lógico). La tabla "vehiculo" gestiona los vehículos con id (clave primaria autoincremental), dominio (obligatorio y único), marca, modelo, anio, seguro_id (foreign key nullable que referencia a seguro_vehicular) y eliminado. La relación entre tablas se estableció mediante FOREIGN KEY con ON DELETE SET NULL, permitiendo que, si se elimina un seguro, el vehículo asociado mantenga sus datos, pero pierda la referencia al seguro, también se crearon índices en los campos de búsqueda frecuente (dominio, marca) y en los campos de estado (eliminado) para optimizar el rendimiento de las consultas.

Orden de Operaciones

Las operaciones tienen un orden, para la inserción de un vehículo con seguro, primero se inserta el seguro en la tabla seguro_vehicular para obtener su ID generado, luego se inserta el vehículo en la tabla vehículo utilizando ese ID como referencia. En las eliminaciones, se actualiza primero el campo eliminado del vehículo y luego el del seguro asociado si existe. Las operaciones de actualización modifican los registros en ambas tablas cuando es necesario, manteniendo la consistencia referencial.

- Reglas de Negocio Principales

Integridad Referencial

La relación entre vehículo y seguro es opcional, permitiendo que existan vehículos sin seguro asociado. Se implementó ON DELETE SET NULL para preservar los vehículos cuando se elimina su seguro asociado. Todas las inserciones y actualizaciones validan la existencia del seguro_id cuando se proporciona.

Unicidad de Datos

El dominio debe ser único por vehículo y el número de póliza único por seguro. Estas validaciones se implementan tanto a nivel de aplicación en la capa Service como a nivel de base de datos mediante constraints UNIQUE, garantizando la consistencia en ambos niveles.

Borrado Lógico

Ningún registro se elimina físicamente de la base de datos. En su lugar, se marca el campo "eliminado" como TRUE. Todas las consultas del sistema incluyen el filtro "WHERE eliminado = FALSE" para excluir los registros inactivos. Esta aproximación permite la recuperación de registros y mantiene el historial completo

❖ Pruebas de Funcionalidad

Se ejecutaron pruebas para cada operación del sistema. Para la creación, se testearon vehículos con y sin seguro, utilizando datos válidos e inválidos para verificar las validaciones. En las operaciones de lectura, se comprobó el listado completo, las búsquedas filtradas por dominio y marca, y la exclusión de vehículos marcados como eliminados. Las actualizaciones incluyeron modificaciones de datos básicos y cambios de seguros asociados. Para las eliminaciones, se verificó el borrado lógico y la no aparición en los listados principales.

Pruebas de Validación: se testearon todos los campos obligatorios, verificando que el sistema rechace entradas vacías. Se validaron los formatos de datos, especialmente para el campo año que debe estar en un rango específico. Se comprobó la unicidad de identificadores como dominio y número de póliza, asegurando que no se permitan duplicados.

Evidencias del Sistema

Menú Principal

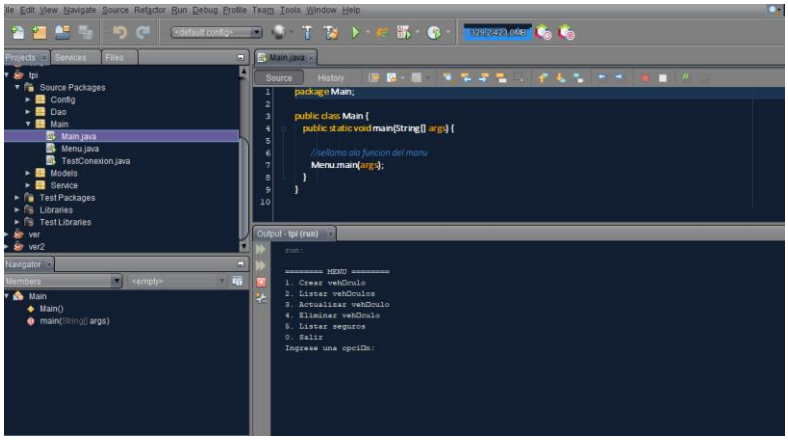


Figura 1: Menú principal del sistema con las 5 opciones disponibles

Operación de Creación

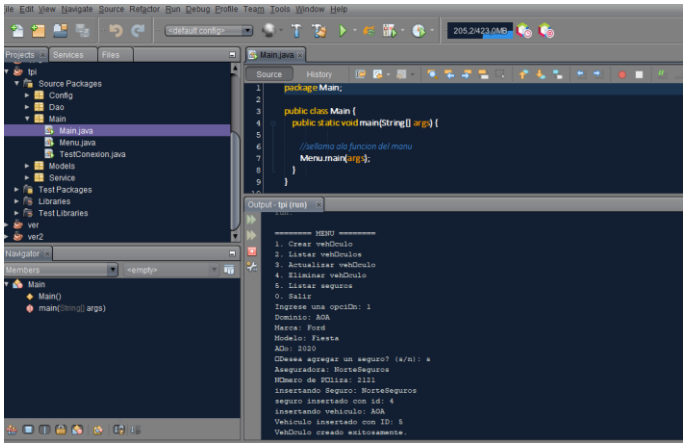


Figura 2: Proceso de creación de vehículo con seguro asociado

Operación de Listado

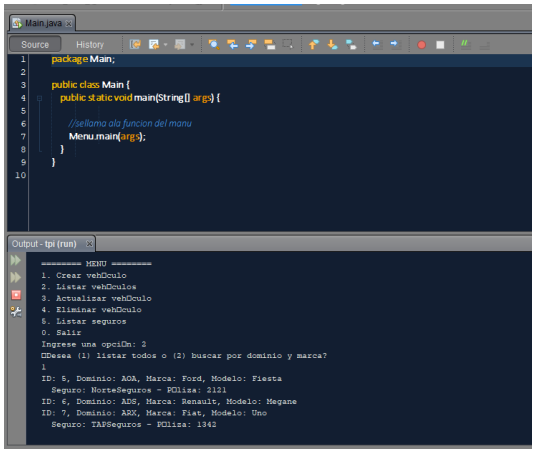


Figura 3: Listado completo de vehículos mostrando seguros asociados

Operación de Búsqueda

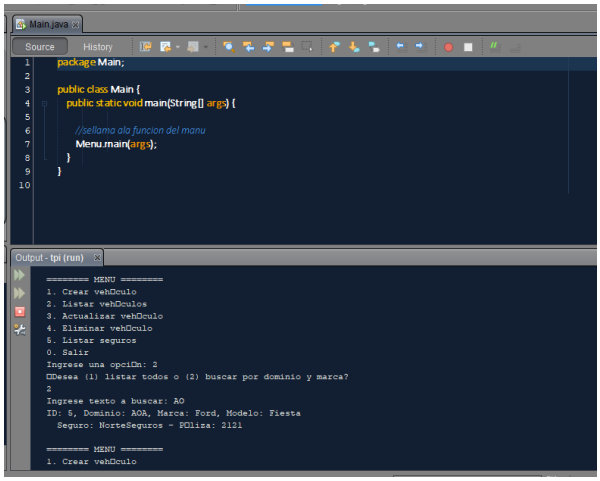


Figura 4: Búsqueda de vehículos por dominio o marca

Operación de Eliminación

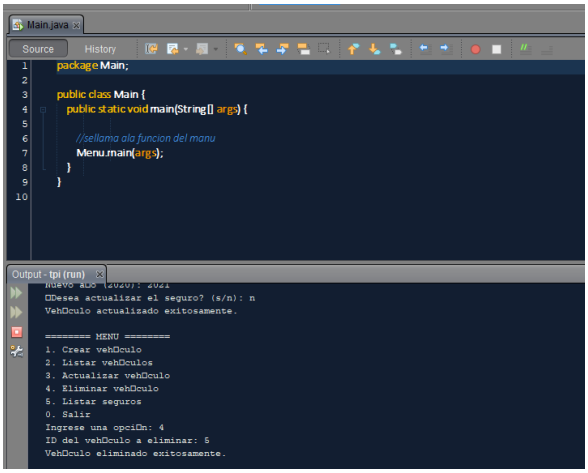


Figura 5: Eliminación lógica de vehículo y verificación en base de datos

Manejo de errores

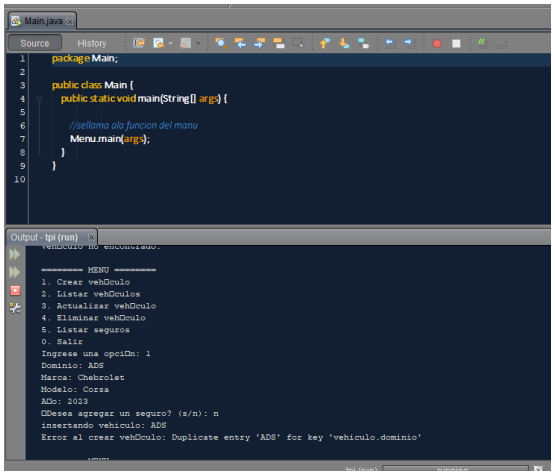


Figura 6: manejo de errores ante entrada invalida

- Conclusiones

El desarrollo de este sistema demostró la efectividad de la arquitectura en capas para facilitar el mantenimiento y la extensión del código. La separación de responsabilidades entre Models, DAO, Service y Config permitió un desarrollo organizado y la posibilidad de modificar componentes sin afectar otros.

La implementación de borrado lógico resultó ser una decisión acertada, ya que preserva el historial completo de registros mientras mantiene la base de datos operativa para consultas diarias. Esta forma de trabajar es especialmente útil en entornos donde la trazabilidad de datos es importante. Las validaciones implementadas en múltiples niveles (aplicación y base de datos) garantizan la integridad de los datos y previeron situaciones de inconsistencia. La relación opcional entre vehículos y seguros proporcionó la flexibilidad necesaria para modelar casos reales donde no todos los vehículos tienen seguro asociado.

Como posibles mejoras pensamos que la incorporación de un módulo de reportes y estadísticas proporcionaría valor agregado al sistema, permitiendo análisis de datos y generación de informes. Un sistema de usuarios y permisos agregaría seguridad y control de acceso a las funcionalidades.

También se podria implementar un historial de cambios para auditoría, notificaciones automáticas por vencimiento de seguros, y la integración con APIs externas para validación de datos de vehículos.