UNIVERSIDAD TECNOLÓGICA NACIONAL

Trabajo Final Integrador (TFI) - Programación 2

Grupo

Grupo146_Vehiculo--SeguroVehicular

Alumnos

Betancort Anahí

Massazza Sergio

Valeria Natalia

Manzanelli López Marcos Gabriel

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Docente Titular

1.Integrantes (4) y roles.

Betancort Anahí – Entities + Config

- Definición de modelado de datos
- Implementación de Vehiculo, SeguroVehicular, Cobertura (enum)
- Desarrollo de DatabaseConnection
- Implementación del TransactionManager

Massazza Sergio – DAO

- Implementación de GenericDAO
- Desarrollo completo de VehiculoDAO y SeguroVehicularDAO
- Mapeo objeto–relacional (JDBC → Entities)
- SQL preparado, inserts transaccionales, joins
- Operaciones CRUD con ResultSet

Manzanelli López Marcos Gabriel – Service

- Implementación de GenericService
- Lógica de negocio: validaciones (dominio, año, póliza, fechas)
- Orquestación transaccional compuesta A+B
- Métodos con commit/rollback
- Reglas de unicidad (dominio y póliza)

Gutierrez Valeria Natalia – Menú (UI por consola)

- Implementación de AppMenu, MenuHandler, MenuDisplay
- Validaciones inmediatas: regex, rangos, fechas
- Funcionalidad completa del CRUD transaccional
- Pruebas funcionales y documentación de resultados

Todos colaboraron en:
✔ Pruebas integradas
✔ Revisión de SQL
✔ Validaciones finales
✔ Integración del proyecto

2. Elección del dominio y justificación

Elegimos el dominio Vehículo → SeguroVehicular de la lista proporcionada en el PDF. Esta opción modela una relación realista en un sistema de gestión de flota vehicular, donde cada vehículo debe tener exactamente un seguro asociado (relación 1→1 obligatoria), pero el seguro no necesita "saber" del vehículo (unidireccionalidad).

Justificación:

Relevancia: Facilita operaciones CRUD compuestas (crear vehículo con seguro en una transacción), validaciones de unicidad (dominio y nroPoliza) y búsquedas por campos clave (dominio para vehículos, nroPoliza para seguros).
Complejidad Técnica: Permite demostrar transacciones (commit/rollback), baja lógica y manejo de FK única en la BD, alineado con los requisitos.
Campos Obligatorios: Coinciden exactamente con el PDF (ej. dominio UNIQUE en Vehículo, nroPoliza UNIQUE en SeguroVehicular, enum Cobertura).
Beneficios: Código extensible para sistemas reales de seguros vehiculares, con validaciones robustas (ej. formato de dominio con regex LLNNNLL).

3. Diseño: decisiones clave

- **Relación 1→1 Unidireccional:** Vehículo (A) contiene una referencia privada a SeguroVehicular (B) (private SeguroVehicular seguro). No hay referencia inversa, garantizando unidireccionalidad pura. La FK se maneja en la capa DAO/Service (no en entidades), con idVehiculo en la tabla SeguroVehicular.
- **FK Única vs. PK Compartida:** Optamos por FK única en la tabla B (idVehiculo UNIQUE, FOREIGN KEY a Vehiculo.id, ON DELETE CASCADE), como recomendado en el PDF. Esto asegura unicidad (un seguro por vehículo) sin compartir PK, simplificando queries JOIN.
- **Arquitectura General:** Patrón DAO para persistencia, capa Service para negocio, menú de consola para UI. Código limpio con excepciones en todas las capas.

4. Arquitectura por capas

El proyecto sigue una arquitectura por capas clara, separando responsabilidades para mantener el código modular y testable.

- **config/:** Gestiona conexiones a BD (DatabaseConnection con método estático getConnection()) y transacciones (TransactionManager con AutoCloseable para commit/rollback automático). Responsable: Conectar a MySQL y manejar sesiones transaccionales.
- **entities/:** Define modelos de dominio (Vehiculo, SeguroVehicular heredan de Base con id y eliminado). Responsable: Representar datos con getters/setters, constructores y toString() legibles.
- **dao/:** Implementa persistencia con JDBC (PreparedStatement en todos los métodos). GenericDAO define contrato; DAOs concretos manejan CRUD, aceptando Connection externa para transacciones. Responsable: Ejecutar queries SQL, mapear ResultSet a objetos, y baja lógica (eliminado = TRUE).
- **service/:** Orquesta negocio y transacciones. GenericService define métodos; Services concretos validan (campos obligatorios, unicidad, formatos) y usan TransactionManager para operaciones compuestas. Responsable: Validaciones, commit/rollback, y reglas 1→1 (impedir múltiples seguros por vehículo).
- **main/:** Punto de entrada (AppMenu.run() con ciclo while). MenuDisplay muestra opciones; MenuHandler procesa entradas con ciclos cerrados (validaciones de formato/unicidad en tiempo real). Responsable: Interfaz de usuario, manejo de errores (parseos, IDs inexistentes) y mensajes claros.

5. Persistencia: estructura, orden y transacciones

5.1 Estructura de tabla Vehiculo

- dominio → UNIQUE
- eliminado → baja lógica

5.2 Estructura de tabla SeguroVehicular

- nroPoliza → UNIQUE
- idVehiculo → FK UNIQUE
- eliminado → baja lógica

5.3 Flujo transaccional

Alta A+B

1. startTransaction()

2. insertar vehículo
3. insertar seguro (usando vehiculoId)
4. commit()
5. catch → rollback()

Actualización

- Actualiza vehículo
- Si hay seguro → actualiza seguro
- Mismo commit/rollback

Eliminación

- Marca vehículo como eliminado
- Marca seguro como eliminado
- 1 sola transacción

6. Validaciones y reglas de negocio

✔ **Dominio**: regex LLNNNLL, mayúsculas, UNIQUE
✔ **Póliza**: obligatoria, mayúsculas, UNIQUE
✔ **Cobertura**: enum (RC, TERCEROS, TODO_RIESGO)
✔ **Año**: 1950 → añoActual+1
✔ **Fecha vencimiento**: debe ser FUTURA
✔ **Baja lógica**: elimina A+B
✔ **Número de chasis**: obligatorio
✔ **Seguro obligatorio al crear vehículo**
✔ **Rollback ante cualquier fallo**

7. Pruebas realizadas

Inicio de la aplicación:

```
+-------------------------------------------------+
|      *** GESTION DE FLOTA VEHICULAR (TPI) ***   |
+-------------------------------------------------+
|                                                 |
|    VEHICULOS (CRUD Compuesto A + B)             |
|-------------------------------------------------|
| 1. Crear Vehiculo (con Seguro, Transaccional)   |
| 2. Listar todos los Vehiculos                   |
| 3. Buscar Vehiculo por ID (con Seguro)          |
| 4. Actualizar Vehiculo (y su Seguro)            |
| 5. Eliminar Vehiculo (Baja Logica A y B)        |
|                                                 |
|    SEGUROS (CRUD Individual B)                  |
|-------------------------------------------------|
| 6. Crear Seguro (para Vehiculo existente)       |
| 7. Actualizar Seguro por ID                     |
| 8. Eliminar Seguro por ID (Baja Logica)         |
| 9. Listar todos los Seguros                     |
|                                                 |
|    BUSQUEDAS POR CAMPO CLAVE                    |
|-------------------------------------------------|
| 10. Buscar Vehiculo por Dominio (Patente)       |
| 11. Buscar Seguro por Nro. de Poliza            |
|                                                 |
+-------------------------------------------------+
| 0. Salir                                        |
+-------------------------------------------------+

  Ingrese una opcion:
```

Creación de vehiculo con seguro:

```
  Ingrese una opcion: 1

--- 1. Crear Vehiculo (Transaccional) ---
--- Validacion de Campos Unicos ---
Paso 1/9 - Dominio (Patente LLNNNLL, ej. AB123CD): AC456AA
Paso 2/9 - Nro. Poliza (Unico): POL431

--- Datos del Vehiculo (A) ---
Paso 3/9 - Marca: Ford
Paso 4/9 - Modelo: Fiesta
Paso 5/9 - Ano (ej. 2024): 2021
Paso 6/9 - Nro. Chasis: CHS-TF-004

--- Datos del Seguro (B) ---
Paso 7/9 - Aseguradora: La Segunda
Cobertura (RC, TERCEROS, TODO_RIESGO): TODO_RIESGO
Paso 9/9 - Fecha Vencimiento (YYYY-MM-DD): 2026-01-01
--- LOG: Transaccion INICIADA (AutoCommit=false) ---
--- LOG: Transaccion finalizada con COMMIT. ---
------------------------------------------------------
EXITO: Vehiculo y Seguro creados (Transaccion OK).
ID Vehiculo: 4 | ID Seguro: 3
------------------------------------------------------
```

Se observa la inserción de datos en ambas tablas:

```
65 •  SELECT v.dominio, v.marca, s.nroPoliza, s.aseguradora
66    FROM vehiculo v
67    JOIN segurovehicular s ON v.id = s.idVehiculo
68    WHERE v.dominio = 'AC456AA';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

dominio	marca	nroPoliza	aseguradora
AC456AA	Ford	POL431	La Segunda

Al duplicar la póliza, el sistema rechaza la operación completa:

```
--- 1. Crear Vehiculo (Transaccional) ---
--- Validacion de Campos Unicos ---
Paso 1/9 - Dominio (Patente LLNNNLL, ej. AB123CD): AF532AB
Paso 2/9 - Nro. Poliza (Unico): POL431
Error: ERROR DE UNICIDAD: Ya existe un seguro activo con la poliza POL431.
Paso 2/9 - Nro. Poliza (Unico):
```

El sistema impide el ingreso de dominios incorrectos:

```
--- 1. Crear Vehiculo (Transaccional) ---
--- Validacion de Campos Unicos ---
Paso 1/9 - Dominio (Patente LLNNNLL, ej. AB123CD): 1234asdf
Error: El formato del dominio es incorrecto. Debe ser LLNNNLL (ej. AB123CD).
Paso 1/9 - Dominio (Patente LLNNNLL, ej. AB123CD):
```

8. Conclusiones y mejoras futuras

El trabajo permitió integrar:

- POO
- Arquitectura en capas
- Transacciones reales
- JDBC
- Validaciones
- Relaciones 1→1
- Manejo de errores

Posibles mejoras

- Interfaz gráfica (JavaFX)
- Logs con Log4j
- Exportación PDF
- Reportes estadísticos
- Tests unitarios

9. Fuentes y herramientas utilizadas

- Oracle. (s.f.). *Java Platform, Standard Edition Documentation*. Oracle. https://docs.oracle.com/en/java
- GitHub, Inc. (s.f.). *GitHub*. https://github.com
- MariaDB Foundation. (s.f.). *MariaDB Documentation*. https://mariadb.org
- MySQL. (s.f.). *MySQL Documentation*. https://dev.mysql.com/doc
- NetBeans. (s.f.). *Apache NetBeans*. https://netbeans.apache.org
- Universidad [UTN]. (2025). *Material de cátedra de [Programación 2]*. [Tecnicatura Universitaria en Programación].
- Gemini https://gemini.google.com/app?hl=es
- ChatGPT https://chatgpt.com/