UNIVERSIDAD TECNOLÓGICA NACIONAL

TECNICATURA EN PROGRAMACIÓN

Trabajo Final Integrador – Programación II

GRUPO 152

Rodrigo Agüero – 35989063

Francisco Alarcón – 42098659

Mauro Zavatti - 37901670

1. Elección del Dominio y Justificación

El dominio seleccionado es el de **Gestión de Pólizas y Vehículos en una Aseguradora**, dando continuidad al modelo desarrollado previamente en Bases de Datos I.

Justificación de la Elección: La elección de este dominio se fundamentó en su idoneidad para cumplir con los objetivos clave de Programación 2:

1. Permite modelar una **relación uno a uno (1→1)** obligatoria entre un Vehiculo y su SeguroVehicular (póliza).
2. Exige la implementación de **transacciones complejas** para garantizar la integridad de los datos, como la creación y eliminación en cascada de pólizas/vehículos.
3. Brinda un entorno real para aplicar una **Arquitectura por Capas** clara, separando la lógica del negocio de las operaciones de acceso a datos, y centralizando la gestión de la conexión y transacciones.

2. Diseño: Decisiones Clave y UML

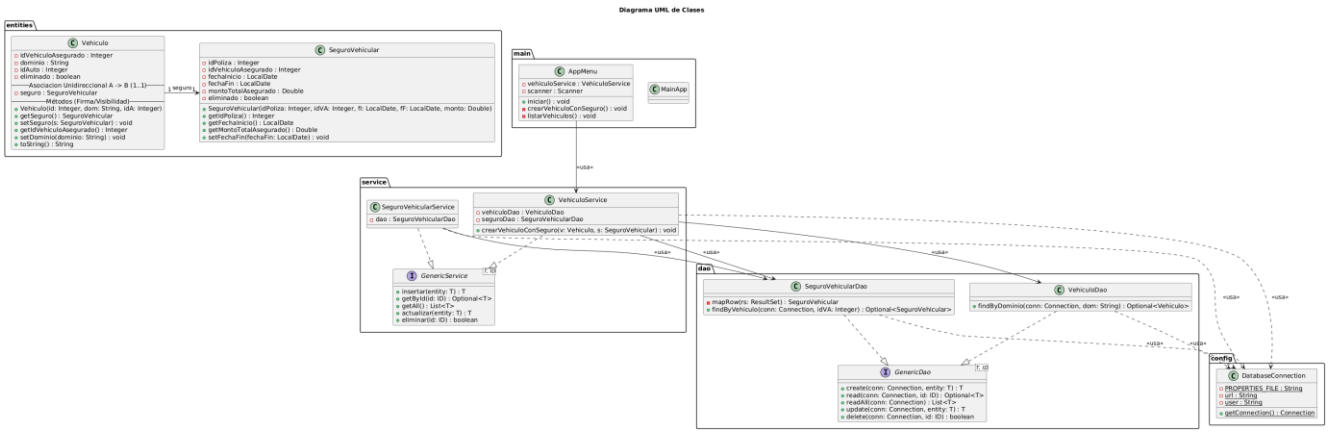
Decisiones Clave de Modelado (Relación 1→1)

- **Entidades Relacionadas:** Vehiculo y SeguroVehicular.
- **Decisión Adoptada:** Se utilizó el patrón de **Foreign Key (FK) Única** para implementar la relación 1→1. La clave foránea (idVehiculoAsegurado) se encuentra en la tabla polizas (entidad SeguroVehicular) y hace referencia a la clave primaria de vehiculos_asegurados (entidad Vehiculo).
- **Justificación:** Se decidió que el campo idVehiculoAsegurado en la tabla polizas fuera una **clave única (UNIQUE)** y a la vez una **Foreign Key (FK)**. Esta elección, si bien requiere un índice adicional, ofrece una semántica clara y robusta: un vehículo existe sin póliza, pero una póliza siempre está asociada a un único vehículo, y esta restricción asegura que solo haya **una** póliza activa en la tabla por vehículo.

Diagrama UML

El diagrama UML de clases muestra la estructura del dominio, destacando las entidades clave (Vehiculo, SeguroVehicular) y la **asociación unidireccional** que se modeló en el código, donde el Vehiculo contiene una referencia a su SeguroVehicular asociado.

- **Clases Principales:**
    - **Vehiculo:** Representa el vehículo asegurado, con su clave primaria (idVehiculoAsegurado), el campo dominio (patente) y la referencia al modelo (idAuto).
    - **SeguroVehicular:** Representa la póliza, con su clave primaria (idPoliza), fechas de vigencia y la clave foránea (idVehiculoAsegurado).



(Se adjunta junto con el trabajo archivo .png con el diagrama)

3. Arquitectura por Capas

El proyecto implementa una arquitectura N-Tier (en capas) para separar las responsabilidades, utilizando el patrón Data Access Object (DAO) y Service Layer.

Paquete/Capa	Descripción de la Capa	Responsabilidad Principal
entities	Capa de Modelo o Dominio	Contiene las clases POJO (Vehiculo, SeguroVehicular) que representan los datos, incluyendo la asociación 1→1.
dao	Capa de Persistencia (DAO)	Se encarga del acceso directo a la base de datos (CRUD). **Recibe la conexión como parámetro**, no la gestiona.
service	Capa de Lógica de Negocio	Contiene la lógica transaccional, las validaciones y las reglas de negocio. **Gestiona la conexión, el COMMIT y el ROLLBACK**.

main	Capa de Presentación	Contiene la interfaz de usuario de consola (AppMenu), que se comunica exclusivamente con la capa Service para ejecutar operaciones.

4. Persistencia: Transacciones y Base de Datos

Estructura de la Base de Datos

- **Motor de Base de Datos:** MySQL (Driver JDBC com.mysql.cj.jdbc.Driver).
- **Nombre de la Base:** aseguradora.
- **Tablas Principales:** vehiculos_asegurados, polizas.

Orden de Operaciones y Transacciones (Commit/Rollback)

Las operaciones de acceso a datos simples usan AutoCommit(true) a nivel de Service. Las operaciones transaccionales complejas se gestionan explícitamente en la **Capa de Servicio** (VehiculoService).

Orden de Operaciones y Transacciones (Commit/Rollback)

Las transacciones complejas, que involucran múltiples operaciones atómicas, se gestionan en la **Capa de Servicio** (VehiculoService).

Ejemplo de Transacción: Eliminación de Vehículo (con Póliza)

La eliminación de un vehículo requiere que su póliza asociada también sea eliminada (Baja Física) para mantener la integridad referencial.

1. **Inicio:** Se obtiene la conexión y se deshabilita el *auto-commit*: conn.setAutoCommit(false);.
2. **Operación 1 (Condicional):** Se busca y se elimina la SeguroVehicular asociada, si existe (seguroDao.delete(conn, s.getIdPoliza())).
3. **Operación 2:** Se elimina el Vehiculo (vehiculoDao.delete(conn, id)).
4. **Confirmación (COMMIT):** Si ambas operaciones son exitosas, se llama a conn.commit();. Esto hace que los cambios sean permanentes.
5. **Reversión (ROLLBACK):** Si una de las operaciones falla (por ejemplo, por un error de BD), se lanza una excepción, se captura el error y se llama a conn.rollback(); antes de cerrar la conexión. Esto revierte todos los cambios de la transacción.

5. Validaciones y Reglas de Negocio

Las validaciones y reglas de negocio se implementan en la **Capa de Servicio** (VehiculoService y SeguroVehicularService) para asegurar que solo los datos válidos y coherentes lleguen a la capa DAO.

Validaciones de Datos (Integridad Básica)

1. **Obligatoriedad de Dominio:** Se valida que el campo dominio (patente) del Vehiculo sea obligatorio y no esté vacío (null o *blank*).
2. **Coherencia Temporal:** Se valida que la fecha de fin de la póliza (fechaFin) debe ser **posterior** a la fecha de inicio (fechaInicio).
3. **Formato de Dominio/Patente:** Se debería validar que el dominio cumpla con el formato de patente vehicular argentino (ej: 6 o 7 caracteres alfanuméricos).
4. **Clave Foránea:** Se verifica que el idAuto asociado al vehículo sea un valor positivo o válido.

Reglas de Negocio (Lógica Aplicada)

1. **Atomicidad de Póliza Inicial:** La creación de un nuevo vehículo y su póliza inicial es una operación atómica obligatoria (crearVehiculoConSeguro).
2. **Unicidad de Póliza:** La base de datos, mediante la restricción UNIQUE/FK, garantiza que un Vehiculo solo puede tener una póliza activa asociada a este campo, impidiendo duplicados.
3. **Duración Máxima de Póliza:** Por norma de la aseguradora, la vigencia de una póliza no puede superar los 365 días (1 año) al momento de la creación.

6. Pruebas Realizadas

Se realizaron pruebas funcionales a través del menú de la consola (AppMenu) para verificar la correcta implementación de las operaciones CRUD y, fundamentalmente, la gestión de transacciones.

Funcionalidades Probadas

- **Crear vehículo + seguro (transacción):** Prueba clave de atomicidad.
- **Eliminar vehículo (transacción):** Prueba de eliminación en cascada, garantizando que el seguro se elimine primero.

- **Actualizar vehículo:** Prueba de persistencia de cambios en la entidad Vehiculo.
- **Listar vehículos con su póliza:** Verificación del *join* en código entre las entidades Vehiculo y SeguroVehicular.

Capturas del Menú y Consultas SQL Útiles

- **Capturas del Menú:** Se incluyeron capturas de pantalla de la ejecución del menú para las opciones 1, 2 y 5 (Crear, Listar, Eliminar), mostrando el mensaje de éxito y el manejo de errores.

  Crear

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 1
Dominio (patente): ABC123
ID de auto (FK a tabla autos): 150019
Fecha inicio p?liza (YYYY-MM-DD): 2025-11-05
Fecha fin p?liza (YYYY-MM-DD): 2026-11-04
Monto total asegurado (opcional, Enter para null): 16500000
? Veh?culo y p?liza creados con ?xito.
```

  Listar

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 2
#2 | CKX064 | id_auto=1 | P?liza ID 3599 | Vigencia 2023-06-06 ? 2024-06-06
#3 | FUT305 | id_auto=2 | (sin p?liza)
#4 | PIV908 | id_auto=3 | (sin p?liza)
#5 | CJS451 | id_auto=4 | P?liza ID 16298 | Vigencia 2013-02-23 ? 2014-02-23
#6 | NAK123 | id_auto=5 | P?liza ID 93904 | Vigencia 2016-05-27 ? 2017-05-27
#7 | BAD197 | id_auto=6 | (sin p?liza)
#8 | LGJ601 | id_auto=7 | (sin p?liza)
#9 | OFJ837 | id_auto=8 | (sin p?liza)
#10 | MAZ251 | id_auto=9 | (sin p?liza)
#11 | NXE869 | id_auto=10 | P?liza ID 3379 | Vigencia 2006-09-12 ? 2007-09-12
#12 | NWP091 | id_auto=11 | P?liza ID 36169 | Vigencia 2022-03-05 ? 2023-03-05
#13 | SKD368 | id_auto=12 | P?liza ID 47472 | Vigencia 2009-01-18 ? 2010-01-18
#14 | HUT286 | id_auto=13 | P?liza ID 17869 | Vigencia 2015-08-16 ? 2016-08-16
#15 | DZC192 | id_auto=14 | P?liza ID 88426 | Vigencia 2015-10-09 ? 2016-10-09
#16 | XLR150 | id_auto=15 | (sin p?liza)
#17 | CIN125 | id_auto=16 | P?liza ID 53062 | Vigencia 2024-12-19 ? 2025-12-19
#18 | JMV058 | id_auto=17 | P?liza ID 57284 | Vigencia 2005-02-26 ? 2006-02-26
#19 | YYM498 | id_auto=18 | (sin p?liza)
#20 | ENC440 | id_auto=19 | P?liza ID 14607 | Vigencia 2005-05-23 ? 2006-05-23
#21 | PIW733 | id_auto=20 | P?liza ID 90945 | Vigencia 2025-07-14 ? 2026-07-14
#22 | VJN021 | id_auto=21 | (sin p?liza)
#23 | AFR939 | id_auto=22 | P?liza ID 52610 | Vigencia 2013-03-19 ? 2014-03-19
#24 | IPU323 | id_auto=23 | (sin p?liza)
#25 | SAA849 | id_auto=24 | P?liza ID 50389 | Vigencia 2025-10-08 ? 2026-10-08
```

```
#150001 | UBZ548 | id_auto=150000 | (sin p?liza)
#150003 | ABC123Z | id_auto=2 | (sin p?liza)
#150004 | ZZZ999 | id_auto=9999 | (sin p?liza)
#150006 | abc123 | id_auto=150002 | P?liza ID 100003 | Vigencia 2025-11-03 ? 2026-11-02
#150007 | ABC123 | id_auto=150005 | P?liza ID 100004 | Vigencia 2025-11-03 ? 2026-11-02
#150010 | ABC123 | id_auto=150019 | P?liza ID 100005 | Vigencia 2025-11-05 ? 2026-11-04
```

Eliminar

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 5
ID de vehiculo_asegurado a eliminar: 150010
? Eliminado correctamente.
```

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 3
ID de vehiculo_asegurado: 150010
No existe un veh?culo con ese ID.
```

Actualizar

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 4
ID de vehiculo_asegurado a actualizar: 150006
Dominio actual: abc123
Nuevo dominio (enter para mantener): XYZ456
id_auto actual: 150002
Nuevo id_auto (enter para mantener):
? Veh?culo actualizado.
```

Buscar

```
=== Men? Veh?culo / Seguro Vehicular ===
1. Crear veh?culo + seguro (transacci?n)
2. Listar veh?culos
3. Buscar veh?culo por ID
4. Actualizar veh?culo
5. Eliminar veh?culo (y su seguro si existe)
X. Salir
Ingrese opci?n: 3
ID de vehiculo_asegurado: 150006
Vehiculo{idVehiculoAsegurado=150006, dominio='XYZ456', idAuto=150002, eliminado=false}
Seguro: SeguroVehicular{idPoliza=100003, idVehiculoAsegurado=150006, fechaInicio=2025-11-03, fechaFin=2026-11-02, montoTotalAsegurado=1.65E7, eliminado=false}
```

- **Consultas SQL Útiles:** Se utilizaron las siguientes consultas SQL para verificar el estado de la base de datos tras las operaciones:

```sql
SELECT v.dominio, v.id_vehiculo_asegurado FROM vehiculos_asegurados v
LEFT JOIN polizas p ON v.id_vehiculo_asegurado = p.id_vehiculo_asegurado
WHERE p.id_poliza IS NULL;
```

```sql
SELECT * FROM polizas
WHERE fecha_fin BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

7. Conclusiones y Mejoras Futuras

Conclusiones (Basadas en la Arquitectura)

1. **Gestión Transaccional Centralizada:** El manejo de transacciones a nivel de **Service Layer** en Java JDBC fue crucial para garantizar la integridad de las operaciones complejas (como la eliminación), evitando registros huérfanos y manteniendo la consistencia de la BD bajo fallas.
2. **Separación de Responsabilidades:** La separación estricta en capas (DAO para el acceso, Service para la lógica) facilitó el desarrollo modular y la reutilización del código, logrando un sistema escalable y fácil de mantener.

Mejoras Futuras (Basadas en el Menú y la Lógica)

1. **Implementar Búsquedas Avanzadas:** Se podría expandir la capa Service para incluir búsquedas más complejas (ej: findByDominio ya existe en DAO, debe exponerse en Service) y búsquedas por rango de fechas de póliza.
2. **Migración a Interfaz Gráfica o Web:** Reemplazar la interfaz de consola (AppMenu) por una interfaz gráfica (Swing/JavaFX) o, preferentemente, una aplicación web, para mejorar la usabilidad y la experiencia del usuario final.

8. Fuentes y Herramientas Utilizadas

Herramientas y Frameworks

Tipo	Nombre
Lenguaje/Entorno	Java 8+ / JDK
IDE	NetBeans
Base de Datos	MySQL (Motor InnoDB)
Conectividad	JDBC (Java Database Connectivity)
Diseño y Modelado	Diagrama UML (Herramienta**:** PlantUML)

Uso de Inteligencia Artificial (IA)

- **Uso de IA:** Sí (Se usó IA para asistencia y optimización del código/documentación).
- **Detalle del Uso:** Utilizada para:
    1. Generación de métodos *boilerplate* (getters/setters, constructores) en las clases de Entidad.
    2. Asistencia en la revisión de la sintaxis y optimización de las sentencias SQL en la capa DAO.
    3. Asistencia en la redacción y estructuración del presente informe técnico.