Enlace video: https://youtu.be/TtWvO4jPkBU

Trabajo Práctico Integrador:
Programación II.

Integrantes: Fausto Gagliano, Daiana Bravo Molins, Pablo de la Puente y Cintia Garcia.

1) Integrantes y Roles

La distribución de roles dentro del equipo se realizó con el objetivo de equilibrar las responsabilidades técnicas, documentales y de coordinación. Se eligió un enfoque colaborativo donde cada integrante aportó desde sus competencias principales, pero también participó en revisiones cruzadas para asegurar coherencia en el producto final. Esta estructura de trabajo facilitó la integración de los módulos y permitió mitigar errores durante las fases de desarrollo y prueba.

El trabajo se distribuyó de la siguiente manera:

- Cintia García – Diseño UML y validaciones. Su aporte permitió que la arquitectura conceptual fuese clara desde el inicio, lo cual redujo inconsistencias durante el desarrollo. También tomó responsabilidad en las pruebas de validación funcional del sistema.

- Daiana Bravo Molins – Programación de los DAOs y consultas SQL, asegurando integridad referencial y correcta interacción entre Java y MySQL.

- Pablo de la Puente – Servicios, manejo de transacciones, documentación formal e integración final. Preparó además la estructura del informe y el material complementario.

- Fausto Gagliano – Coordinación general, armado de entidades, interacción con el menú principalmente revisión general

:

2) Elección del Dominio y Justificación

Se eligió el dominio de pacientes e historias clínicas porque representa un caso realista, estructurado y altamente regulado en el mundo profesional. La relación 1→1 entre Paciente e Historia Clínica es un ejemplo clásico de identificación y atributos extendidos, lo cual lo vuelve ideal para aplicar conceptos de integridad referencial y reglas de negocio.

Además, trabajar con datos médicos exige validaciones estrictas, formato uniforme y reglas claras, lo cual permitió demostrar:

• Uso adecuado de claves únicas (DNI, número de historia clínica).

• Borrado lógico para preservar trazabilidad histórica.

• Transacciones que garanticen consistencia en inserts compuestos.

• Justificación realista de por qué un paciente solo puede tener una única historia clínica activa.

Este dominio permitió conectar las exigencias académicas con escenarios verosímiles del mundo profesional.

3) Diseño: Decisiones Clave (1→1, FK única vs PK compartida) + UML

Una de las principales decisiones de diseño fue cómo modelar la relación 1→1 en la base de datos. Tras analizar las dos alternativas recomendadas (FK única en B, o PK compartida), se eligió la clave foránea única en Historia Clínica porque ofrece:

• Mayor flexibilidad para extender atributos sin obligar a que ambas tablas compartan PK.

• Independencia en la generación de identificadores.

• Compatibilidad con ORMs y frameworks comunes en proyectos reales.

• Mayor claridad al leer la estructura SQL.

El UML definió de forma clara la dirección de la asociación: Paciente conoce su Historia Clínica, pero la historia clínica no necesita conocer al paciente desde el código Java. Esto reduce el acoplamiento y simplifica la serialización e impresión
en consola.

El diseño también contempló atributos obligatorios, uso de enums (Grupo Sanguíneo), métodos equals() basados en id, y encapsulamiento estricto.

4) Arquitectura por Capas – Responsabilidades

La arquitectura se diseñó siguiendo un enfoque multicapa clásico, lo cual facilita el mantenimiento, la escalabilidad
y las pruebas. Cada capa tiene responsabilidades bien delimitadas:

- config/: responsable de centralizar la conexión a la base de datos. Permite reutilizar el métodogetConnection() sin repetir código y simplifica futuros cambios (credenciales, motor de BD, pool de conexiones, etc.).

- entities/: define el modelo del dominio con atributos privados, constructores, getters y setters.Incluye lógica mínima para evitar sobrecargas (SRP) y mantiene el sistema fácilmente testeable.

- dao/: implementa el acceso directo a MySQL mediante JDBC. Cada DAO usa PreparedStatementpara evitar inyecciones SQL y para mejorar la eficiencia en consultas recurrentes. Los métodos reciben una conexión externa cuando participan de transacciones.

- service/: concentra lógica de negocio, validaciones, comprobaciones de unicidad y manejo explícito de transacciones (commit / rollback). Esta capa es fundamental para asegurar la coherencia del sistema.

- main/: capa de presentación basada en consola. No accede directamente a la base, sino que se comunica con los servicios.

Esto garantiza separación estricta de responsabilidades.

5) Persistencia: Estructura de la Base, Orden de Operaciones y Transacciones

La estructura de la base se diseñó siguiendo criterios de normalización, integridad referencial y seguridad. Las decisiones clave fueron:

• Uso de claves primarias autoincreméntales para ambas entidades.

• HistoriaClinica posee paciente_id con UNIQUE, garantizando la relación 1→1.

• ON DELETE CASCADE asegura que si un paciente se elimina físicamente (solo a efectos de prueba), su historia clínica también se elimine, manteniendo integridad.

Respecto a las transacciones, se decidió:

1. Abrir la conexión desde Service mediante setAutoCommit(false).

2. Encadenar operaciones (por ejemplo crear Paciente → crear HistoriaClinica).

3. Evaluar éxito de cada operación parcial.

4. Si todo es correcto: commit().

5. Si alguna operación falla: rollback() inmediato.

6. Finalmente restaurar autoCommit(true) para no afectar operaciones aisladas.

Este flujo garantiza atomicidad y evita inconsistencias como historias clínicas sin paciente asociado.

6) Validaciones y Reglas de Negocio

Las validaciones son esenciales en este dominio por su sensibilidad y naturaleza crítica. Se implementaron las siguientes reglas de negocio:

- El DNI debe ser único y no vacío.
- El número de historia clínica debe ser único.
- Un paciente no puede poseer más de una historia clínica.
- La fecha de nacimiento debe ser válida (no futura).
- Los campos obligatorios como nombre, apellido y grupo sanguíneo deben estar completos.
- Eliminado funciona como borrado lógico (permite conservar registros históricos).

Además, desde service se verifica que no existan violaciones de unicidad antes de ejecutar transacciones, así evitando errores SQL y mejorando la experiencia del usuario en el menú.

7) Pruebas Realizadas

Se ejecutaron múltiples rondas de pruebas, entre ellas:

- Pruebas de inserción con valores válidos.

• Pruebas de inserción con DNI duplicado para verificar control de excepciones.

• Pruebas de creación de historia clínica sin paciente (debe fallar).

• Pruebas de actualización de datos sensibles como grupo sanguíneo y antecedentes.

• Pruebas de eliminación lógica y listado filtrado.

• Pruebas de transacción: se forzó un error intencional durante el guardado compuesto y se verificó que ninguna tabla recibiera datos parciales (rollback correcto).

También se realizaron consultas SQL para verificar la integridad referencial, índices, unicidad y cascadas.

8) Conclusiones y Mejoras Futuras

El proyecto demuestra alineación total con los objetivos académicos del TFI: arquitectura en capas, uso correcto del patrón DAO, aplicación de transacciones, manejo de excepciones, validaciones y modelado adecuado de relaciones 1→1.
Entre las mejoras futuras propuestas:

• Migrar a una interfaz gráfica en Swing o JavaFX.

• Incorporar autenticación por roles (médico, recepcionista, administrador).

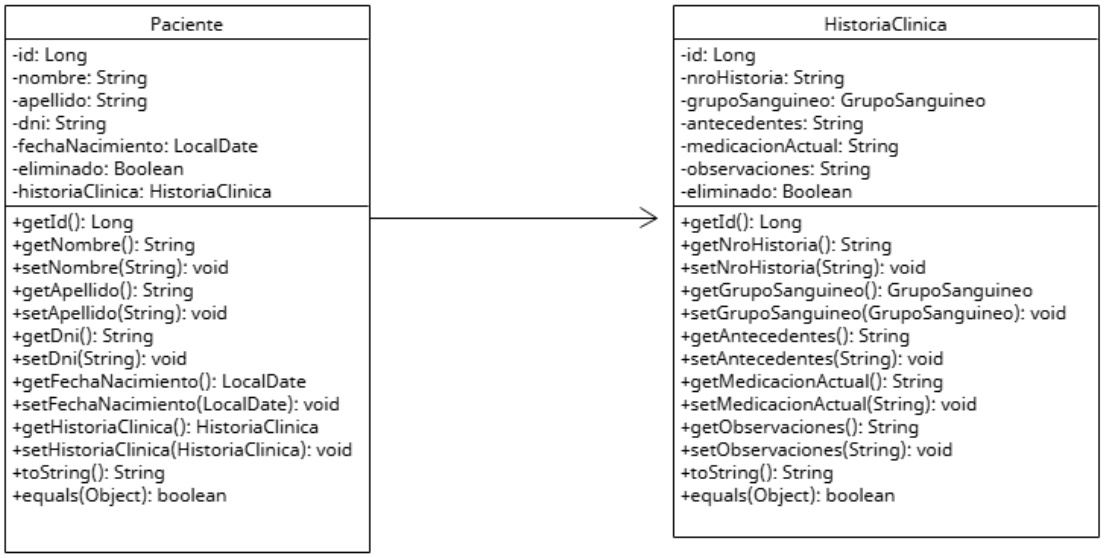
• Agregar historial de modificaciones para auditoría.

• Implementar API REST para integrarlo con otros sistemas.

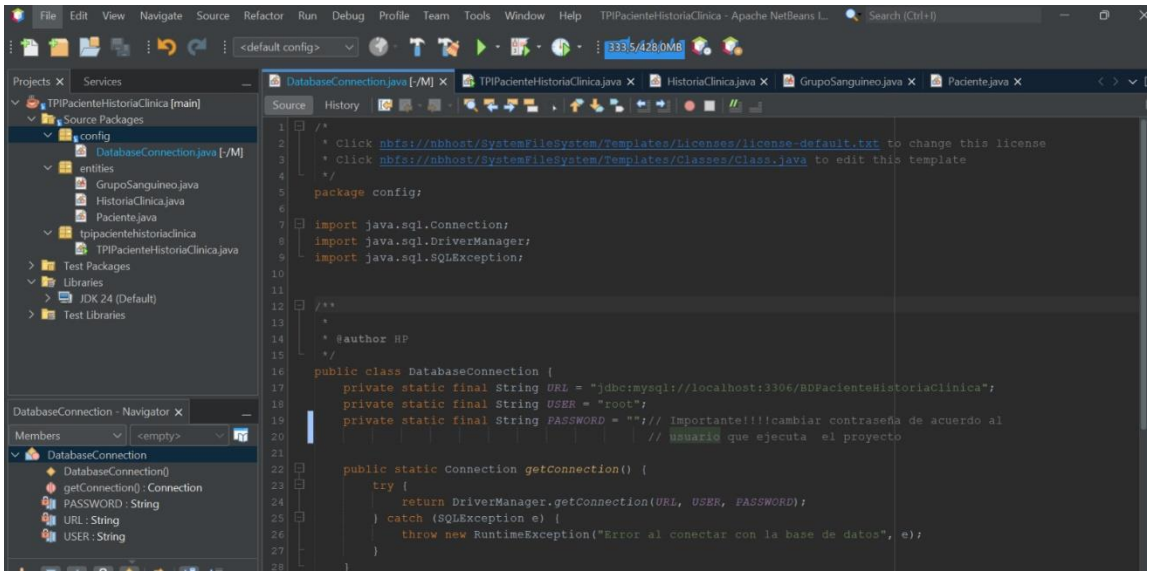
• Migrar a un ORM como Hibernate para reducir código repetitivo en los DAOs.

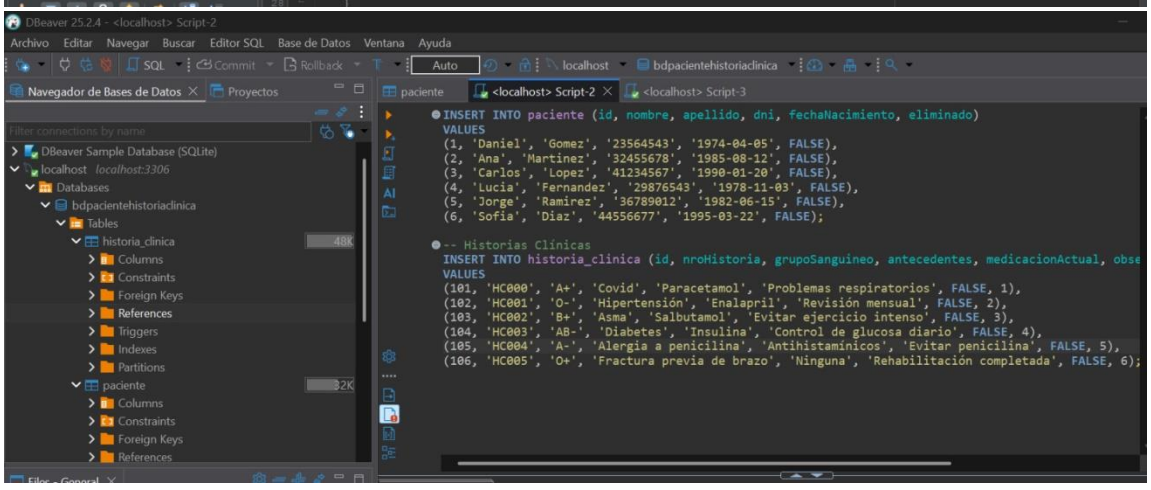
Imágenes del Programa:

- Diagrama UML

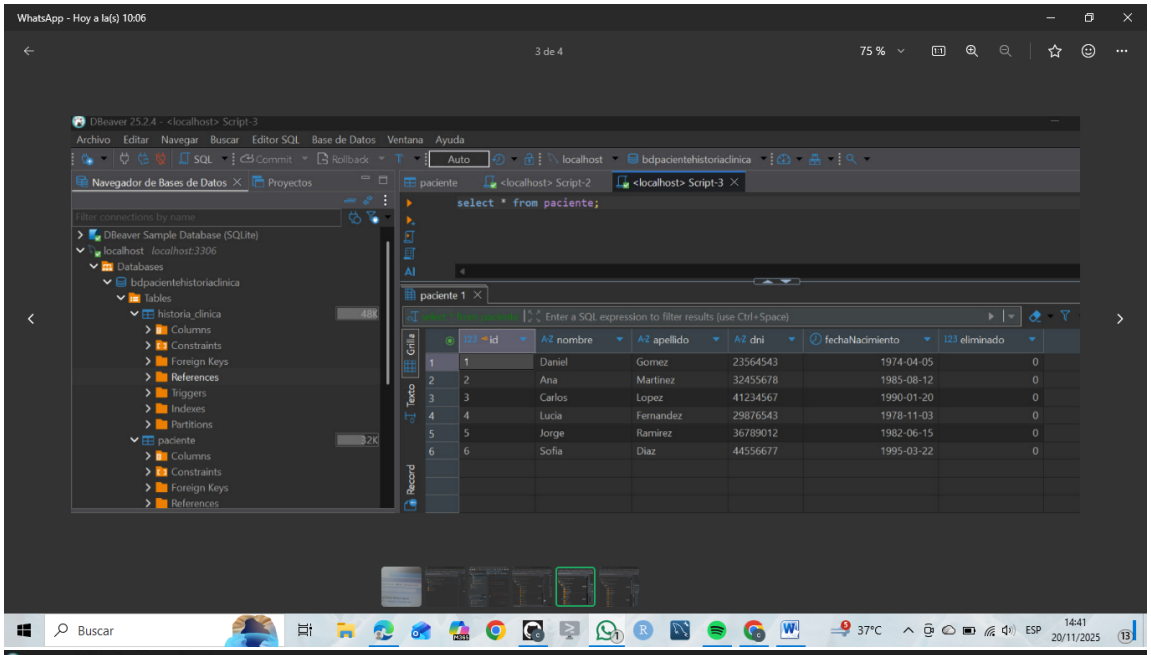


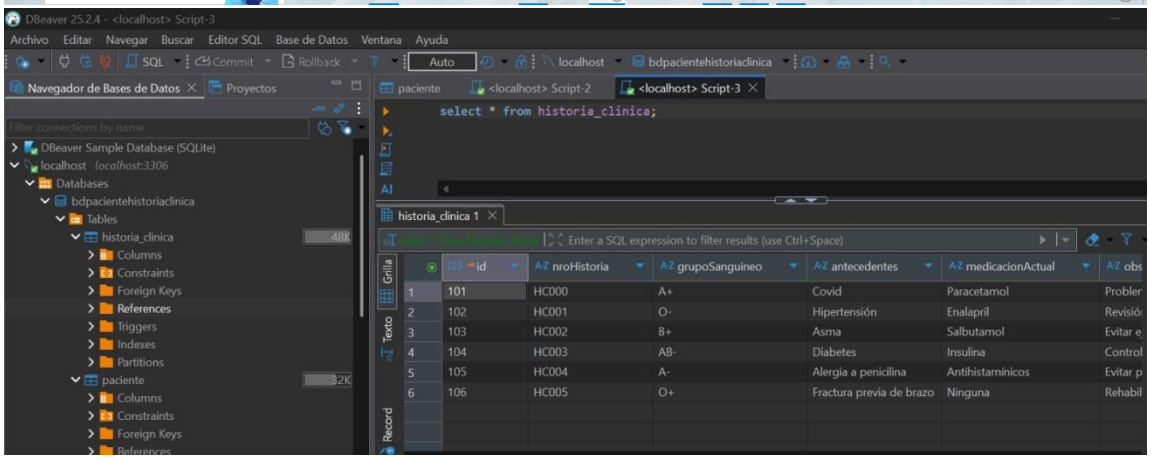
- Algunas pantallas de NetBeans:

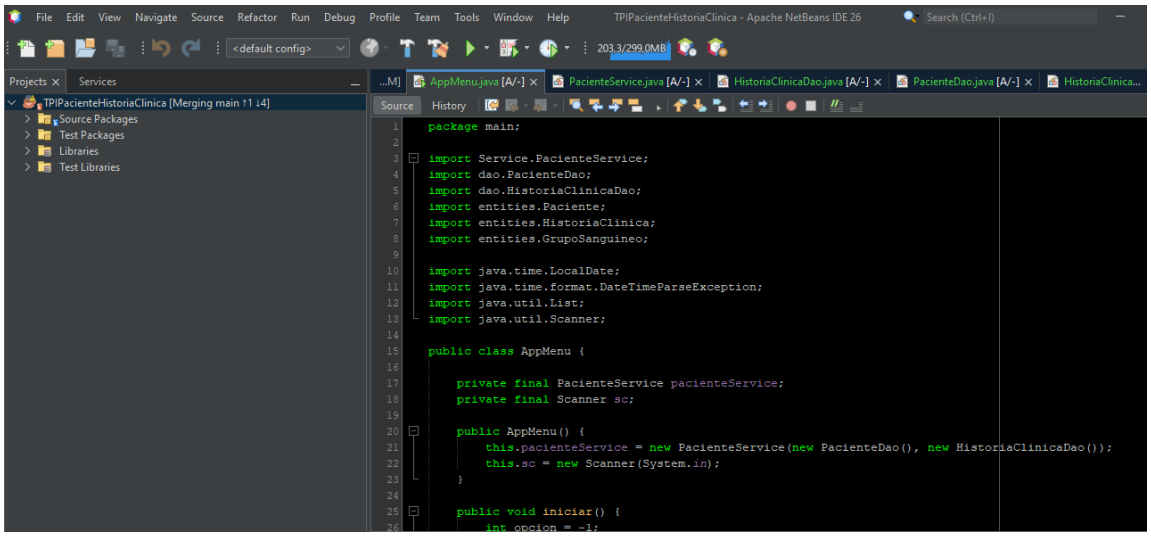


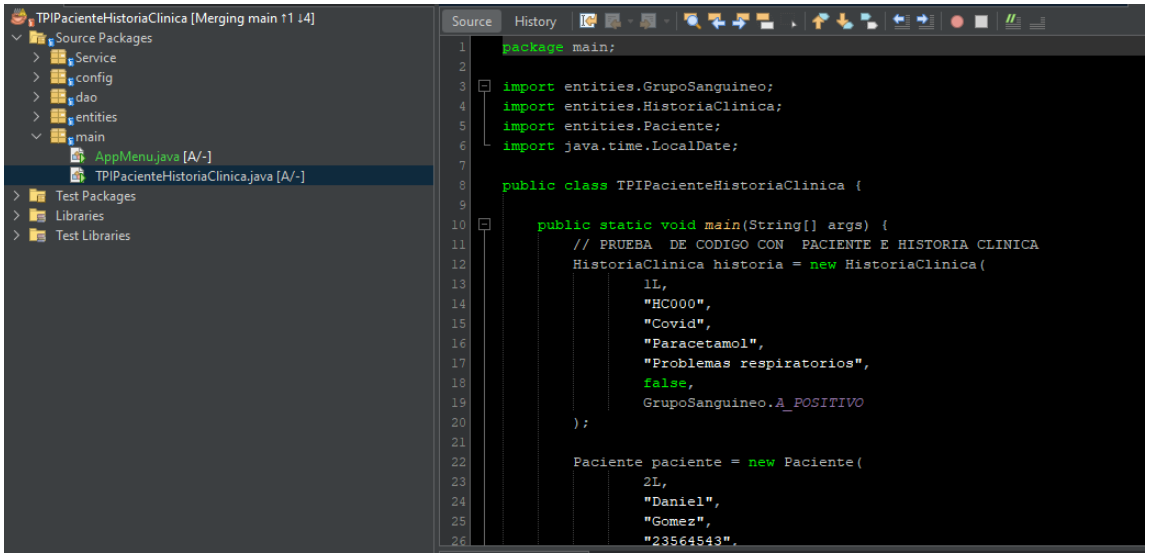


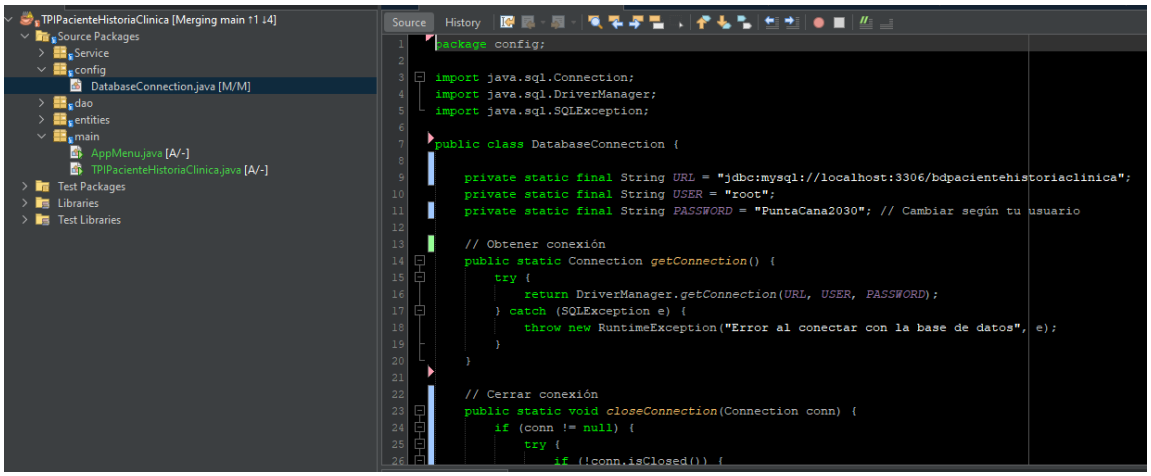
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA TRABAJO
PRACTICO INTEGRADOR

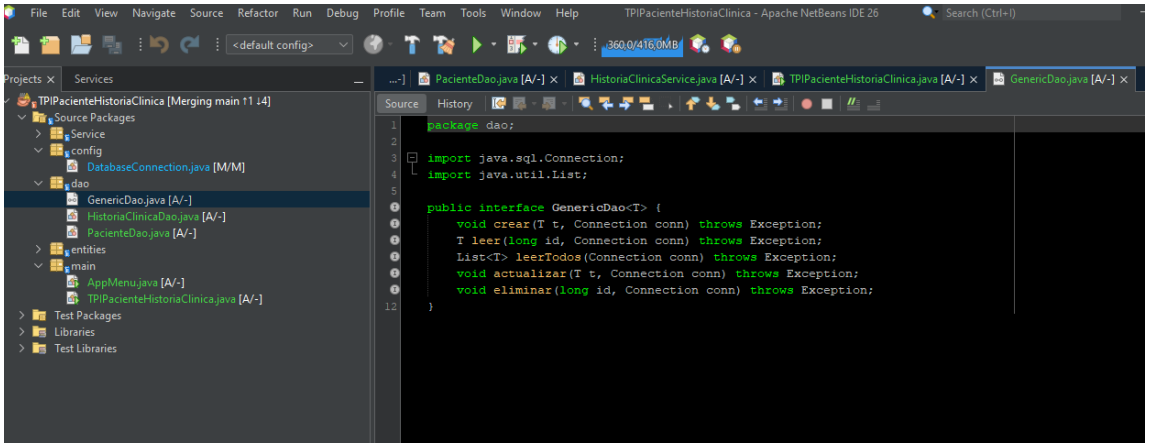








- 

- 

9) Fuentes y Herramientas Utilizadas

• Java 17 – Lenguaje principal.

• MySQL – Motor de base de datos.

• JDBC – Acceso a datos.

• NetBeans – IDE utilizado.

• GitHub – Control de versiones y repositorio.

• ChatGPT – Asistencia en documentación y redacción.

Programación TPI