Trabajo Final Integrador - Programación II

Alumnos del grupo 90:

- Martin Eluney Gomez Piñeiro
- Pedro Nicolás Hidalgo
- Joaquin Edgar Escobar
- Fiorella García Galfione

Tema: Libro - FichaBibliografica

Profesor: Ariel Enferrel

Profesor ayudante: Renzo Sosa

1. Introducción y Roles

1.1. Integrantes y Roles

Nombre	Rol Principal	Tareas Clave
Joaquin y Fiorella	Capa de Servicios y Transacciones	Diseño de la lógica de LibroService, gestión de commit/rollback, validaciones de negocio, y desarrollo del Menú de Consola (AppMenu).
Martin	Desarrollador DAOs	Implementación de GenericDAO y LibroDao/FichaBibliograficaDAO, asegurando la inyección de Connection.
Pedro	Desarrollador de Entidades y BBDD	Desarrollo de las clases Libro y FichaBibliografica, diseño y creación de la BBDD (SQL)
Fiorella	Documentación, diagrama UML y Pruebas Unitarias	Generación del informe y el diagrama UML, y ejecución de pruebas de integración (TestConexion).

1.2. Elección del Dominio y Justificación

El dominio elegido para la resolución del TPI es **Libro → FichaBibliografica**.

Esta elección se justifica por su clara aplicación de la relación **uno a uno (1:1)**. En el contexto de una biblioteca, cada libro (objeto físico o digital) debe tener exactamente una ficha bibliográfica que contenga su metadato único (ISBN, clasificación Dewey, ubicación, etc.). Esta pareja de clases cumple con la premisa de que la clase "A" (Libro) referencia a la clase "B" (FichaBibliografica) de forma única e ineludible.

2. Diseño de Arquitectura y Patrones

2.1. Arquitectura por Capas

El proyecto se implementó siguiendo la arquitectura tradicional por capas para asegurar la separación de responsabilidades, promoviendo la legibilidad y el mantenimiento:

Paquete/Capa	Responsabilidad	Dependencias
config	Gestión de la conectividad y lectura de propiedades (db.properties).	Ninguna (Capa base).
edu.utn.entities	Modelado del dominio (Libro, FichaBibliografica) y contención de datos.	Ninguna (Capa de dominio).
DAO (Data Access Object)	Acceso a la base de datos, mapeo de objetos a SQL y viceversa. Utiliza JDBC.	→ edu.utn.entities, → java.sql
service	Lógica de negocio, validaciones y orquestación de transacciones.	→ DAO, → config, → edu.utn.entities
edu.utn	Punto de entrada y capa de presentación por consola (Main, AppMenu).	→ service

2.2. Diseño UML y Decisiones Clave

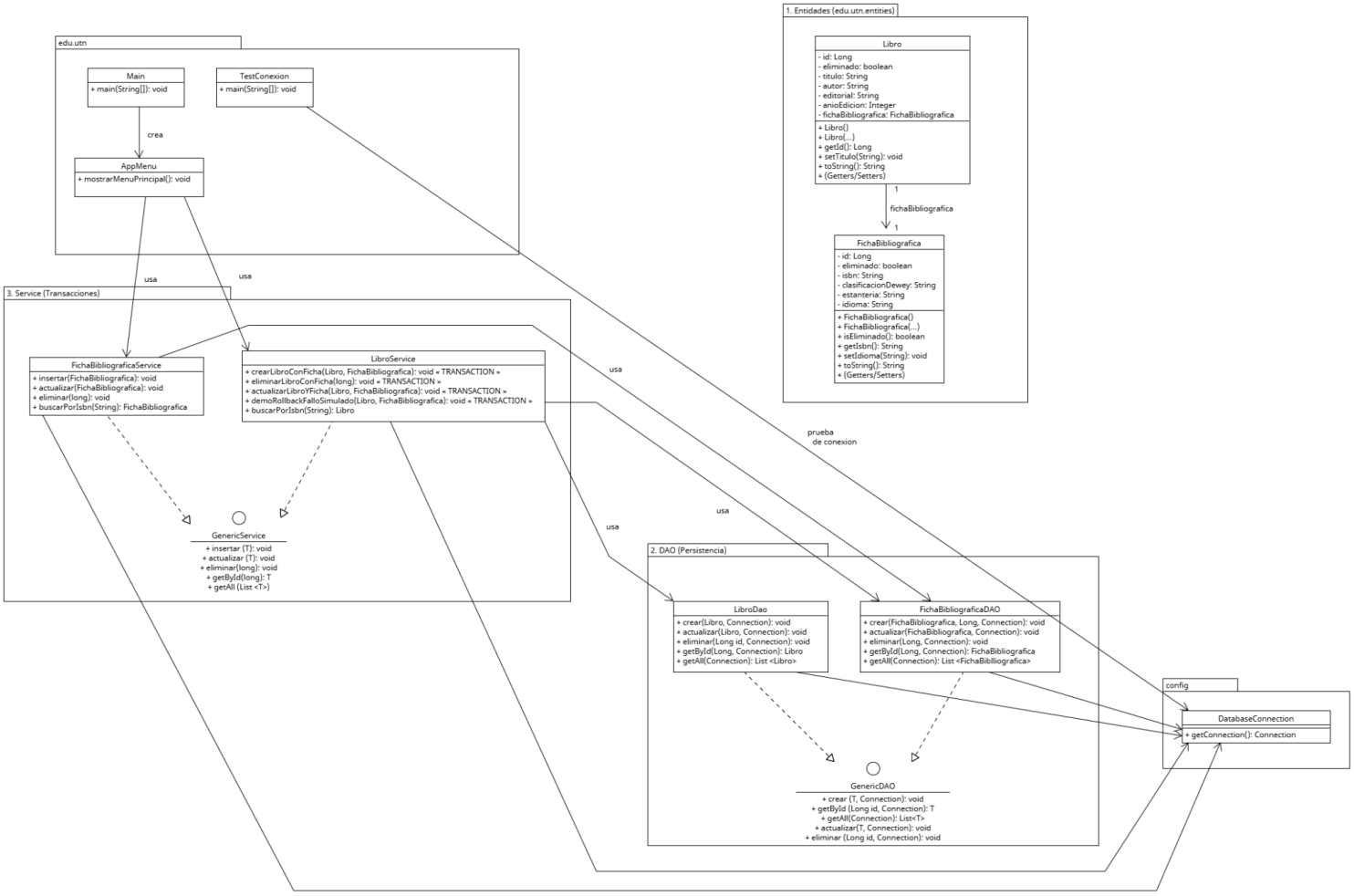
Relación 1:1 Unidireccional

Se implementó una asociación 1 a 1 unidireccional (Libro → FichaBibliografica), cumpliendo con la consigna de que solo la clase "A" (Libro) contiene el atributo que referencia a "B" (fichaBibliografica).

Persistencia de la Relación (Decisión Clave)

- Opción Elegida: Clave Foránea Única en la Tabla de B (fichas_bibliograficas).

- Implementación SQL: La tabla fichas_bibliograficas contiene la columna id_libro, la cual es tanto una FOREIGN KEY referenciando a libros.id como una restricción UNIQUE KEY.
- Justificación: Esta es la forma más limpia y estándar de forzar la unicidad de la relación (1:1) a nivel de la base de datos, garantizando que un Libro solo pueda tener una FichaBibliografica asociada.



3. Persistencia y Transacciones

3.1. Orden de Operaciones Compuestas

Las operaciones de crear y actualizar son compuestas y se deben ejecutar en un orden específico para satisfacer la integridad referencial:

- Operación INSERT (Crear):
    1. Insertar el Libro en la tabla libros.
    2. Recuperar el ID generado (LAST_INSERT_ID()) del nuevo Libro.
    3. Insertar la FichaBibliografica en la tabla fichas_bibliograficas, utilizando el ID recuperado en el paso 2 como clave foránea (id_libro).

- Operación UPDATE (Actualizar):
    1. Actualizar los campos de la tabla libros por ID.
    2. Actualizar los campos de la tabla fichas_bibliograficas por ID (o por id_libro).
- Operación DELETE Lógico:
    1. Actualizar el campo eliminado = TRUE en la tabla libros. (Opcionalmente, se puede replicar la baja lógica en fichas_bibliograficas).

3.2. Gestión Transaccional (commit / rollback)

La gestión transaccional reside exclusivamente en la Capa de Servicios (LibroService.java), asegurando que las operaciones compuestas (Libro + Ficha) se realicen de forma atómica.

Acción	**Lugar de Ejecución (Clase/Método)**	**Propósito**
conn.setAutoCommit(false)	LibroService.crearLibroCompleto() (al inicio)	Desactiva el modo por defecto para iniciar la transacción.
conn.commit()	LibroService.crearLibroCompleto() (dentro del try)	Confirma los cambios realizados por ambos DAOs si todas las operaciones fueron exitosas.
conn.rollback()	LibroService.crearLibroCompleto() (dentro del catch de SQLException)	Deshace todos los cambios realizados desde el setAutoCommit(false) si ocurre algún error (ej. violación de clave única por ISBN).

conn.setAutoCommit(true)	LibroService.crearLibroCompleto() (dentro del finally)	Restablece la conexión a su estado original antes de cerrarla.

3.3. Uso de PreparedStatement

Todos los métodos CRUD en la capa DAO (LibroDao y FichaBibliograficaDAO) utilizan java.sql.PreparedStatement para:

- Prevenir ataques de inyección SQL.
- Mejorar la eficiencia al reutilizar consultas.

4. Validaciones y Reglas de Negocio

Las validaciones se implementaron en la Capa de Servicios (LibroService) para interceptar errores de negocio antes de la persistencia:

Regla de Negocio	**Implementación en LibroService**
ISBN Único	Delegada a la Base de Datos (UNIQUE KEY en fichas_bibliograficas). Si se intenta crear un ISBN duplicado, el LibroService captura la SQLException y ejecuta ROLLBACK.
Relación 1:1	Delegada a la Base de Datos (UNIQUE KEY en id_libro). El crearLibroCompleto verifica que el ID del Libro se asigne correctamente a la Ficha como FK.
Campos Obligatorios	Validación de nulidad o vacío (.isEmpty()) del título y la existencia de la FichaBibliografica al inicio del método crearLibroCompleto().

Baja Lógica	Los métodos eliminar en el DAO solo actualizan el campo eliminado = TRUE y el Service lo ejecuta transaccionalmente. La lógica de negocio (getAll) filtra automáticamente los registros con eliminado = TRUE.

5. Pruebas Realizadas

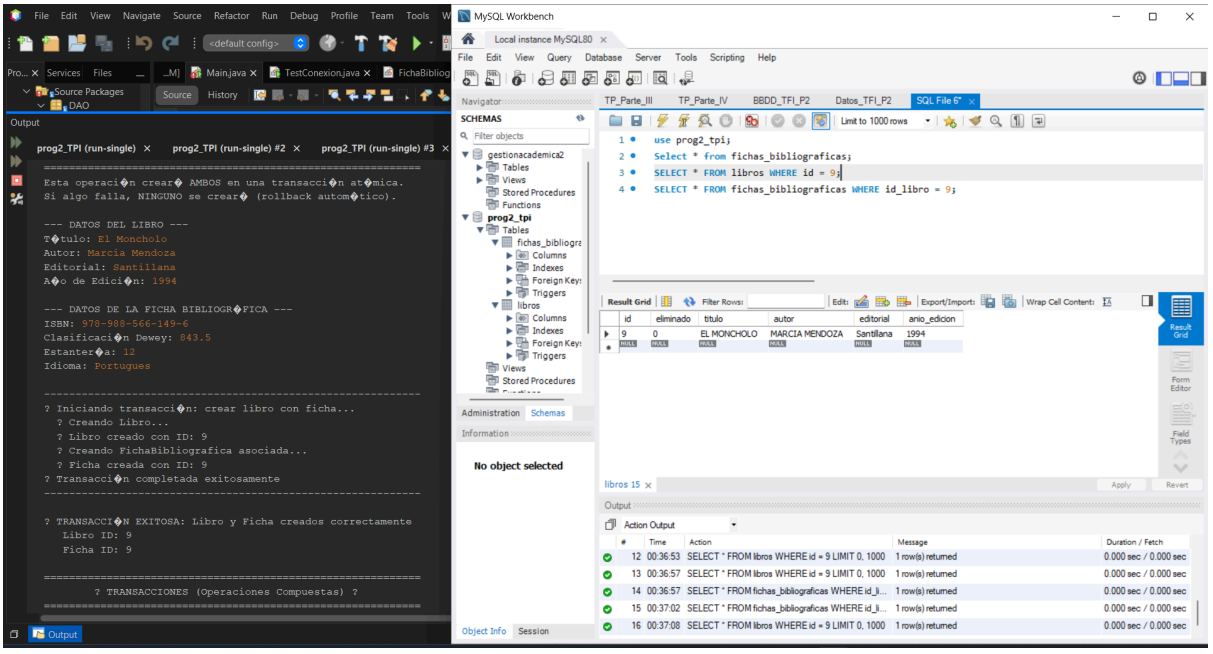
Se realizaron pruebas de integración y transaccionales a través del menú de consola (AppMenu).

5.1. Prueba de Conectividad (TestConexion)

- Resultado: Se comprobó la conexión exitosa a la BBDD antes de las operaciones CRUD, verificando la correcta configuración de db.properties y la disponibilidad del driver JDBC.

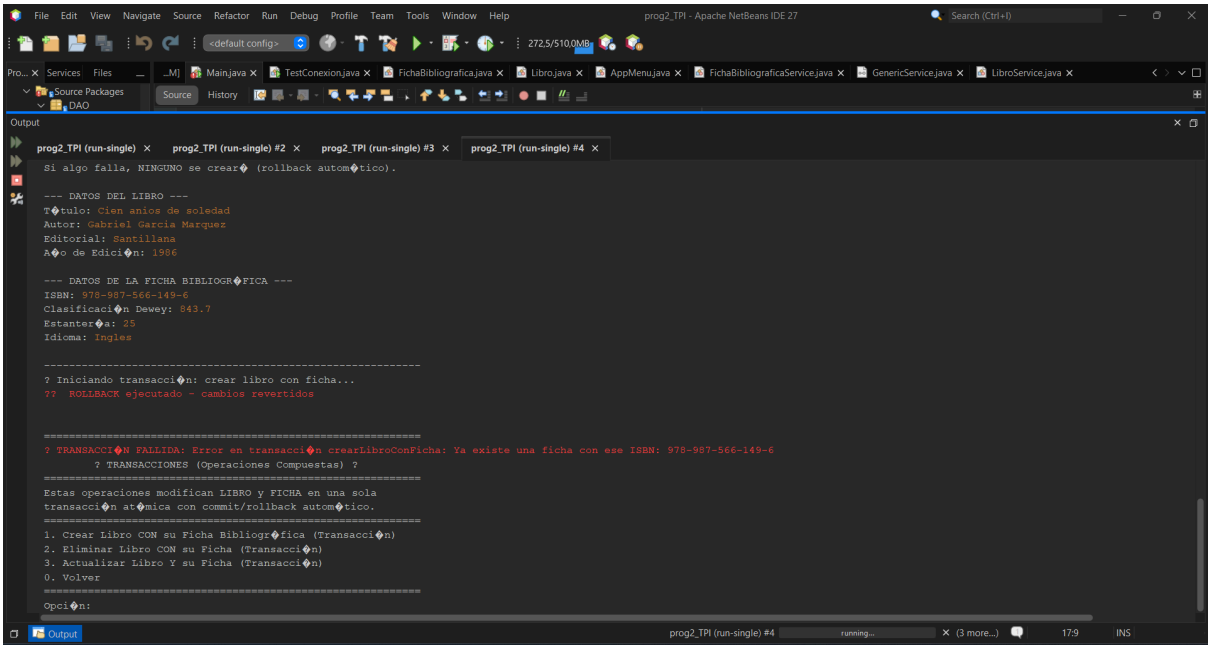
5.2. Prueba de Transacción Exitosa (CREATE - Commit)

- Acción: Ingreso de un nuevo libro con un Título y un ISBN únicos.
- Resultado Esperado: Mensaje "Libro y Ficha creados exitosamente (TRANSACCIÓN COMMIT)".
- Verificación SQL: SELECT * FROM libros WHERE id = [nuevo ID]; y SELECT * FROM fichas_bibliograficas WHERE id_libro = [nuevo ID]; deben mostrar los registros.



5.3. Prueba de Transacción Fallida (CREATE - Rollback)

- Acción: Intentar ingresar un nuevo libro con un ISBN que ya existe en la base de datos (Violación de Clave Única).
- Resultado Esperado: El LibroService captura la SQLException, ejecuta el ROLLBACK, e imprime un mensaje de error detallado en la consola.
- Verificación SQL: Se comprueba que no se haya insertado ningún registro en la tabla libros y fichas_bibliograficas, demostrando que el ROLLBACK evitó la inserción parcial del Libro sin su Ficha asociada.



6. Conclusiones y Mejoras Futuras

6.1. Conclusiones

El proyecto cumple satisfactoriamente con todos los requisitos del TPI: la relación 1:1 unidireccional está implementada tanto en el código Java como en el diseño de la BBDD (FK única), y la capa de servicios maneja las transacciones compuestas de forma robusta, garantizando la atomicidad de las operaciones CRUD.

6.2. Mejoras Futuras

1. Mapeo de Ficha en Lectura: Mejorar los métodos getById y getAll del LibroDao para que realicen un JOIN y mapeen automáticamente el objeto FichaBibliografica dentro del objeto Libro.
2. Manejo de Excepciones: Crear clases de excepciones personalizadas (ej. LibroNotFoundException, DatabaseAccessException) para mejorar la claridad de los errores en la capa AppMenu.
3. Configuración: Implementar un patrón Singleton en DatabaseConnection para gestionar mejor las conexiones.

7. Referencias

- Herramientas de Desarrollo: Java Development Kit (JDK), NetBeans/VS Code.
- Base de Datos: MySQL Server (o MariaDB) con MySQL Connector/J.
- Patrones de Diseño: Data Access Object (DAO), Service Layer.
- Asistencia:
    - Asistencia para el diseño arquitectónico del UML.

8. Video

Video TPI