Trabajo Final Integrador (TFI)

Materia: Programación 2

Coordinador: Carlos Martinez
Profesores: Ariel Enferrel, Cinthia Rigoni, Alberto Cortez

Grupo: Número 83

Integrantes:

Nogueira Damián
Galarza Manuel
Etchegoyen Gabriel

- Índice interactivo (cada ítem lleva al marcador con el tema elegido)

- Integrantes y divisiones de tareas en general – Grupo 83

Manuel Galarza

- Diseño del dominio y UML
- Creación del modelo relacional y scripts SQL
- Pruebas en MySQL
- Edición video

Gabriel Etchegoyen

- Implementación de Entities y DAO
- Validaciones de campos
- Documentación técnica del acceso a datos
- Armado PDF y documentación

Damián Nogueira

- Implementación de Services
- Manejo de transacciones y rollback
- Desarrollo de AppMenu y conclusiones
- Estructura y armado de proyecto en repositorio

- **Elección del Dominio: Libro → FichaBibliográfica**

Optamos por el dominio Libro - FichaBibliográfica por las siguientes razones:

1. Relación 1→1 Natural: Cada libro físico posee una única ficha bibliográfica que contiene información catalográfica específica, estableciendo una relación uno a uno perfectamente definida.

2. Contexto Real y Familiar: El ámbito bibliotecario es conocido y proporciona un contexto tangible para implementar las operaciones CRUD, facilitando la comprensión del modelo.

3. Atributos Específicos y Relevantes:
   - Libro: campos como título, autor, editorial y año de edición.
   - FichaBibliográfica: datos técnicos cómo ISBN (único), clasificación Dewey, estantería e idioma.

4. Validaciones Significativas:
   - Formato y unicidad del ISBN.
   - Validación del año de edición.
   - Control de duplicados en títulos/autor.

5. Búsqueda por Campo Relevante: La implementación de búsqueda por “ISBN” resulta particularmente útil y realista en este contexto, siendo un identificador estándar en el ámbito bibliotecario.

6. Estructura Clara para Transacciones: Las operaciones de alta (crear libro + ficha) y baja (eliminación lógica de ambos) se prestan naturalmente para implementar transacciones que garanticen la consistencia de los datos.

Este dominio permite demostrar efectivamente todos los requisitos técnicos del trabajo integrador mientras se mantiene un modelo coherente y fácil de comprender.

- Diseño: Decisiones Clave + UML

- Decisiones Clave de Diseño:

1. Relación 1→1 Unidireccional

- Implementación: La clase Libro contiene una referencia a FichaBibliografica (-fichaBibliografica : FichaBibliografica)
- Unidireccionalidad: Solo Libro conoce a FichaBibliografica, no viceversa
- Ventaja: Simplifica el modelo y evita dependencias circulares

2. Estrategia de Clave Foránea: FK Única vs PK Compartida

Decisión: FK Única en FichaBibliografica

- Razones:

Flexibilidad: Permite que ambas entidades tengan sus propias secuencias de ID
Mantenimiento: Facilita operaciones individuales sobre FichaBibliografica
Claridad: La relación queda explícita en la base de datos mediante libro_id UNIQUE
Consistencia: Mantiene el patrón estándar de entidades independientes

3. Diseño de Entidades

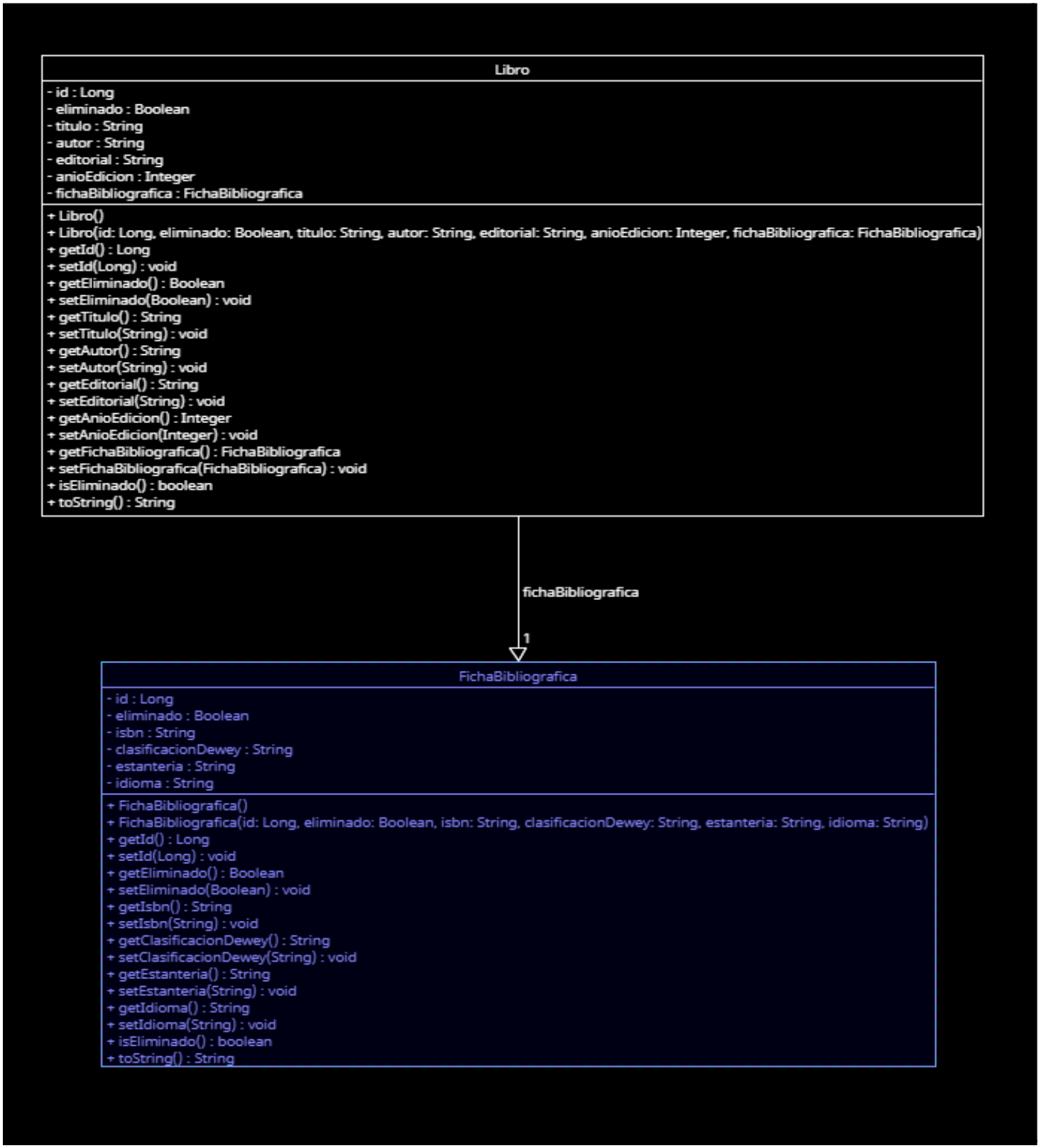
Atributos coherentes: Cada clase contiene atributos específicos de su dominio
Baja lógica: Ambas implementan eliminado para soft delete
Constructores: Incluyen constructor vacío (reflexión) y completo
Métodos auxiliares: isEliminado() y toString() para funcionalidad adicional

Ventajas del Diseño Elegido:

- Separación de responsabilidades: Información básica del libro vs datos catalográficos
- Extensibilidad: Fácil agregar nuevos campos a cualquiera de las entidades
- Normalización: Cumple con principios de diseño de bases de datos
- Mantenibilidad: Cambios en una entidad no afectan necesariamente a la otra
- Performance: Consultas eficientes mediante índices en isbn y libro_id

Este diseño garantiza la integridad de la relación 1→1 mientras mantiene la flexibilidad para operaciones individuales y futuras expansiones del modelo.

- Diagrama UML generado para este proyecto:



- Arquitectura de Capas del Sistema de Gestión de Libros y Fichas Bibliográficas

Este trabajo final integrador implementa una aplicación Java por capas que gestiona libros y sus fichas bibliográficas, manteniendo la relación 1–1 entre ambas entidades. El sistema se conecta a una base de datos MySQL y demuestra conceptos de persistencia, validación, transacciones y atomicidad.

- Estructura de Capas:

- Capa: Entidad (Modelo)
Paquete: LibroFicha_entities
Descripción: Define las clases Libro y FichBibliografía con sus atributos y métodos.

- Capa: DAO (Acceso a Datos)
Paquete: LibroFicha_dao
Descripción: Implementa operaciones CRUD mediante PreparedStatement.

- Capa: Service (Lógica de Negocio)
Paquete: LibroFicha_service
Descripción: Aplica validaciones y maneja transacciones (commit/rollback).

- Capa: Configuración
Paquete: LibroFicha_config
Descripción: Contiene DatabaseConnection que lee las credenciales de db.properties.

- Capa: Presentación
Paquete: LibroFicha_main
Descripción: Interfaz de consola (AppMenu y Main).

- Capa: Integración BD I
Paquete: SeguridadBD.integracionBDI
Descripción: Clases del TFI de Bases de Datos I usadas para consultas seguras y DTOs (únicamente referenciales).

- Persistencia de datos

Estructura de la bases de datos:

```sql
-- Conexión al servidor se hace con:
-- mysql -u tu_usuario -p < tpi_libro_ficha_schema.sql

-- === Reiniciar todo desde cero ===
DROP DATABASE IF EXISTS tpi_libro_ficha;
CREATE DATABASE tpi_libro_ficha CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE tpi_libro_ficha;

-- === Tabla: libro (lado A) ===
CREATE TABLE libro (
  id           BIGINT AUTO_INCREMENT PRIMARY KEY,
  eliminado    BOOLEAN NOT NULL DEFAULT FALSE,
  titulo       VARCHAR(150) NOT NULL,
  autor        VARCHAR(120) NOT NULL,
  editorial    VARCHAR(100),
  anioEdicion  INT
) ENGINE=InnoDB;

-- (Índices sencillos para búsquedas)
CREATE INDEX idx_libro_titulo ON libro (titulo);
CREATE INDEX idx_libro_autor  ON libro (autor);

-- === Tabla: ficha_bibliografica (lado B) ===
CREATE TABLE ficha_bibliografica (
  id                 BIGINT AUTO_INCREMENT PRIMARY KEY,
  eliminado          BOOLEAN NOT NULL DEFAULT FALSE,
  isbn               VARCHAR(17) NOT NULL,
  clasificacionDewey VARCHAR(20),
  estanteria         VARCHAR(20),
  idioma             VARCHAR(30),
  libro_id           BIGINT NOT NULL,
  CONSTRAINT uq_ficha_isbn  UNIQUE (isbn),       -- cada ISBN una sola vez
  CONSTRAINT uq_ficha_libro UNIQUE (libro_id),   -- relación 1:1 con libro
  CONSTRAINT fk_ficha_libro FOREIGN KEY (libro_id)
    REFERENCES libro(id)
    ON DELETE CASCADE
) ENGINE=InnoDB;
```

- Tabla libro (Entidad A)

id: BIGINT AUTO_INCREMENT PRIMARY KEY
eliminado: BOOLEAN NOT NULL DEFAULT FALSE (baja lógica)
titulo: VARCHAR(150) NOT NULL
autor: VARCHAR(120) NOT NULL
editorial: VARCHAR(100)
anioEdicion: INT

- Índices: idx_libro_titulo, idx_libro_autor para optimizar búsquedas

- Tabla ficha_bibliografica (Entidad B)

id: BIGINT AUTO_INCREMENT PRIMARY KEY
eliminado: BOOLEAN NOT NULL DEFAULT FALSE (baja lógica)
isbn: VARCHAR(17) NOT NULL UNIQUE
clasificacionDewey: VARCHAR(20)
estanteria: VARCHAR(20)
idioma: VARCHAR(30)
libro_id: BIGINT NOT NULL

- Restricciones de Integridad

UNIQUE uq_ficha_isbn: Garantiza que cada ISBN sea único en el sistema
UNIQUE uq_ficha_libro: Implementa la relación 1→1 (un libro tiene una única ficha)
FOREIGN KEY fk_ficha_libro:

- Referencia a libro(id)
ON DELETE CASCADE: Eliminación en cascada para mantener consistencia

- Orden y de operaciones

Estrategia implementada: A → B (Libro primero, Ficha después)

- Operación de CREACIÓN (Libro + Ficha)

Método: insertarConFicha()

Orden ejecutado:
conn.setAutoCommit(false) - Inicio de transacción
libroDao.crear(conn, libro) - Insertar LIBRO (entidad A)
ficha.setLibroId(libroId) - Vincular FICHA con LIBRO mediante FK
fichaDao.crear(conn, ficha) - Insertar FICHA BIBLIOGRÁFICA (entidad B)
conn.commit() - Confirmar transacción completa

Secuencia: A → B (Libro primero, Ficha después)

- Operación de ELIMINACIÓN (Libro + Ficha)

Método: eliminar(Long id)

Orden ejecutado:
conn.setAutoCommit(false) - Inicio de transacción
fichaDao.eliminarPorLibroId(conn, id) - Eliminar FICHA BIBLIOGRÁFICA (entidad B)
libroDao.eliminar(conn, id) - Eliminar LIBRO (entidad A)
conn.commit() - Confirmar transacción completa

Secuencia: B → A (Ficha primero, Libro después)

- Transacciones (Dónde se hace commit/rollback)

Estrategia de Manejo de Transacciones

En la capa Service se implementa el control transaccional mediante el patrón try-catch-finally:

Inicio de Transacción

conn.setAutoCommit(false);

Ubicación: Al inicio de cada método transaccional (crear, actualizar, eliminar)

Confirmación (Commit)

conn.commit();

Ubicación: Al final del bloque `try`, después de todas las operaciones exitosas

Reversión (Rollback)

rollbackSilencioso(conn);

Ubicación: En el bloque `catch`, cuando ocurre cualquier excepción

Limpieza de Recursos

restaurarYCerrar(conn);

Ubicación: En el bloque `finally`, garantizando la ejecución siempre

Métodos de Soporte Transaccional

Rollback Silencioso:
- Maneja excepciones durante el rollback sin propagarlas
- Evita fallos en cascada si la conexión está corrupta

Restauración y Cierre:
- Restablece `autoCommit = true` para conexiones futuras
- Cierra la conexión de forma segura

- Flujo Transaccional por Operación

El manejo de transacciones se implementa en la capa Service mediante un patrón try-catch-finally. Para operaciones simples (insertar, actualizar) se inicia la transacción con setAutoCommit(false), se ejecuta la operación y se confirma con commit(). Si ocurre algún error, se ejecuta rollbackSilencioso().

Para operaciones compuestas (eliminar, insertarConFicha) se agrupan múltiples operaciones en una misma transacción. En eliminación se elimina primero la Ficha y luego el Libro; en creación se crea primero el Libro y luego la Ficha. El commit solo se ejecuta si todas las operaciones son exitosas, garantizando atomicidad.

- Validaciones

```java
// ============================================================
// VALIDACIONES DE NEGOCIO (simples y claras)
// ============================================================

/**
 * Valida un Libro antes de persistirlo/actualizarlo.
 * Reglas mínimas: título y autor obligatorios; año dentro de un rango razonable.
 */
private void validarLibro(Libro libro) {
    if (libro == null) throw new IllegalArgumentException("Libro no puede ser null");

    if (libro.getTitulo() == null || libro.getTitulo().isBlank()) {
        throw new IllegalArgumentException("Título obligatorio");
    }
    if (libro.getAutor() == null || libro.getAutor().isBlank()) {
        throw new IllegalArgumentException("Autor obligatorio");
    }

    // Año opcional, pero si viene, debe estar en rango [1450..año actual].
    if (libro.getAnioEdicion() != null) {
        int y = libro.getAnioEdicion();
        int current = Year.now().getValue();
        if (y < 1450 || y > current) {
            throw new IllegalArgumentException("Año de edición fuera de rango (1450.." + current + ")");
        }
    }
}

/**
 * Valida una FichaBibliografica. Para este TFI el ISBN es obligatorio y
 * validamos un formato simple: ISBN-10 o ISBN-13 (permitimos guiones/espacios).
 */
private void validarFicha(FichaBibliografica ficha) {
    if (ficha == null) throw new IllegalArgumentException("Ficha no puede ser null");

    String raw = (ficha.getIsbn() == null ? "" : ficha.getIsbn()).replaceAll("[- ]", "");
    boolean isbn10 = raw.matches("\\d{9}[\\dX]"); // 10 (el último puede ser X)
    boolean isbn13 = raw.matches("\\d{13}");      // 13 dígitos
    if (!(isbn10 || isbn13)) {
        throw new IllegalArgumentException("ISBN debe ser ISBN-10 (10) o ISBN-13 (13) válido");
    }
}
```

- Validación de Libro (Entidad A)

Libro no nulo
Título obligatorio (no null, no vacío)
Autor obligatorio (no null, no vacío)
Año de edición opcional, pero si se proporciona debe estar entre 1450 y año actual

- Validación de Ficha (Entidad B)

Ficha no nula
ISBN obligatorio
Formato ISBN válido: ISBN-10 (9 dígitos + 1 dígito o 'X') o ISBN-13 (13 dígitos)
Se permiten guiones y espacios en el ISBN (se limpian antes de validar)

- Reglas de relación 1 → 1

Integridad referencial garantizada por FK única en base de datos
Ficha debe tener libroId no nulo
Operaciones atómicas en Service para mantener consistencia

- Puntos de validación

Validaciones se ejecutan antes de cada operación transaccional
Implementadas en capa Service (no en DAO)
Excepciones: IllegalArgumentException con mensajes descriptivos

- Pruebas realizadas

Pruebas del Menú de Consola

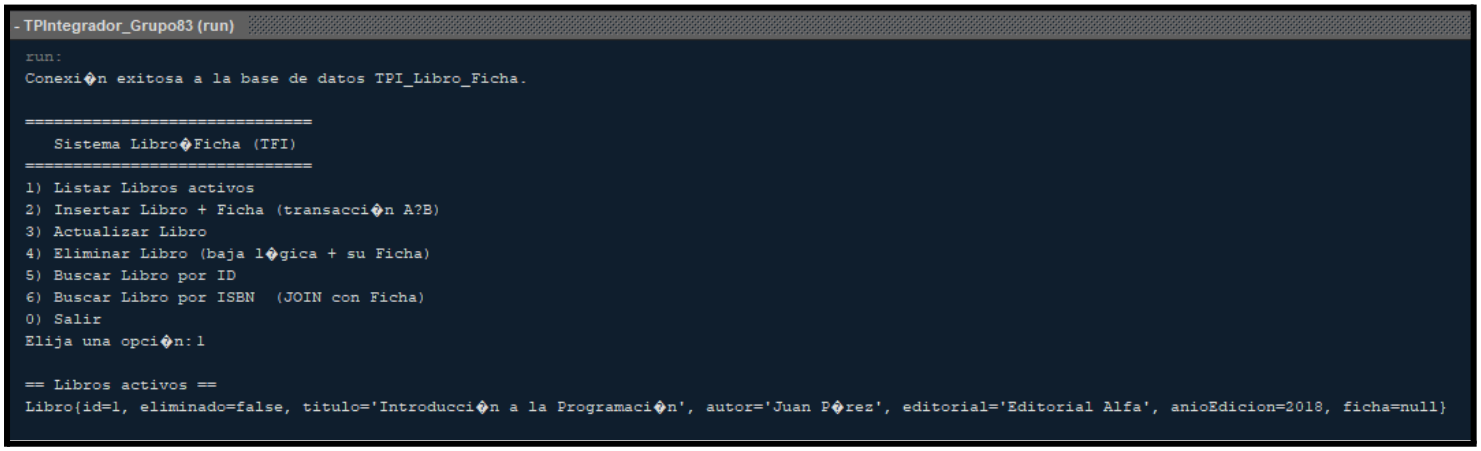
Acción ejecutada: Opción 1 del menú - "Listar Libros activos"

Resultado observado:

Conexión exitosa a la base de datos TPI_Libro_Ficha

Interfaz clara con todas las opciones del CRUD disponibles

Funcionalidades verificadas:
✅ Conexión a base de datos establecida
✅ Menú de consola funcional
✅ Navegación entre opciones operativa
✅ Mensajes de estado claros para el usuario



- Se probó el sistema con el libro "El Libro de la Selva":

- Libro: Título, autor, editorial, año 1894
- Ficha: ISBN 978-0141325293, clasificación Dewey, estantería, idioma

Prueba exitosa:
- Creación transaccional (Opción 2 del menú)
- Validaciones de ISBN y año funcionando
- Relación 1→1 establecida correctamente
- Búsqueda por ISBN operativa

```
==============================
   Sistema Libro?Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci?n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l?gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci?n: 2

== Insertar Libro + Ficha (transacci?n A?B) ==
T?tulo (obligatorio):Libro de la selva
Autor (obligatorio):Rudyard Kipling
Editorial (opcional):Macmillan Publishers
A?o de edici?n (opcional, ENTER para omitir)1894
ISBN (obligatorio, 10/13 d?gitos, admite guiones)978-0141325293
Clasificaci?n Dewey (opcional):823.8
Estanter?a (opcional):h4
Idioma (opcional):Español
```

- Volvemos a listar resultados de libros activos

```
==============================
   Sistema Libro?Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci?n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l?gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci?n: 1

== Libros activos ==
Libro{id=1, eliminado=false, titulo='Introducci?n a la Programaci?n', autor='Juan P?rez', editorial='Editorial Alfa', anioEdicion=2018, ficha=null}
Libro{id=5, eliminado=false, titulo='Libro de la selva', autor='Rudyard Kipling', editorial='Macmillan Publishers', anioEdicion=1894, ficha=null}
```

- Actualización del año y título (a modo de ejemplo) del libro agregado

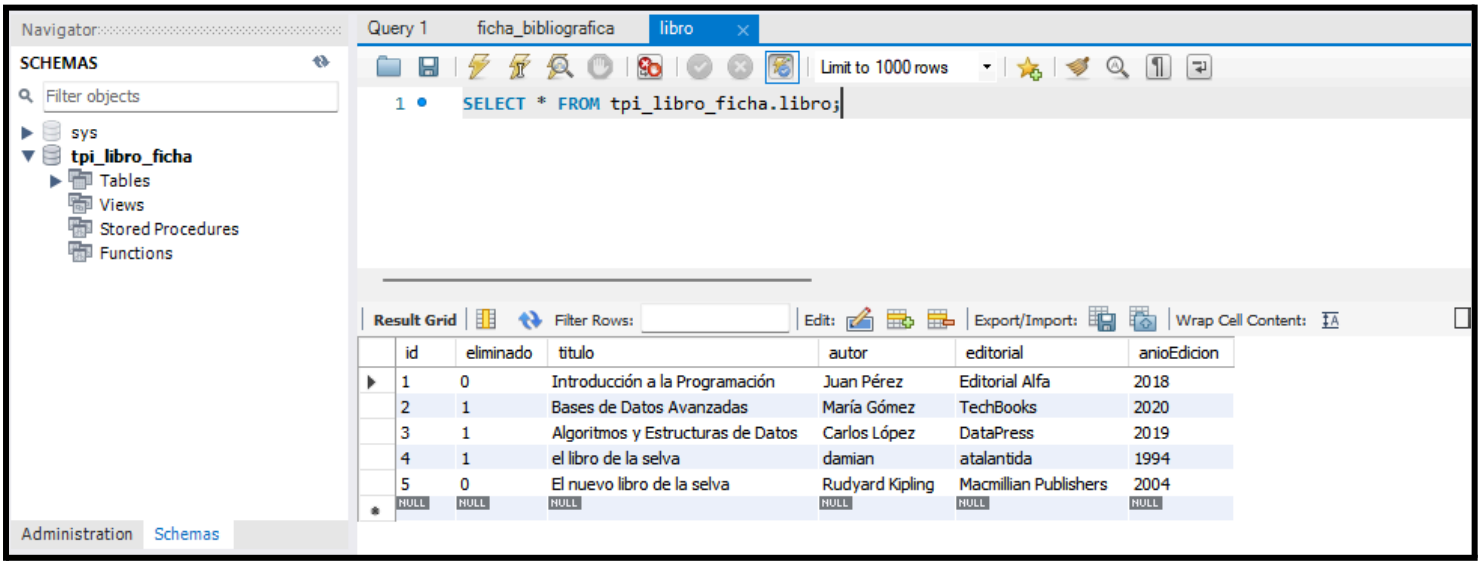
```
== Libros activos ==
Libro{id=1, eliminado=false, titulo='Introducci?n a la Programaci?n', autor='Juan P?rez', editorial='Editorial Alfa', anioEdicion=2018, ficha=null}
Libro{id=5, eliminado=false, titulo='Libro de la selva', autor='Rudyard Kipling', editorial='Macmillan Publishers', anioEdicion=1894, ficha=null}

==============================
   Sistema Libro?Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci?n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l?gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci?n: 3

== Actualizar Libro ==
ID del libro a actualizar:5
```

- Se corrige título y año para simular actualización de un libro elegido por su ID (5)

```
t - TPIntegrador_Grupo83 (run)
  Elija una opci▯n: 3

  == Actualizar Libro ==
  ID del libro a actualizar:5
  Valores actuales: Libro{id=5, eliminado=false, titulo='Libro de la selva', autor='Rudyard Kipling', editorial='Macmillan Publishers', anioEdicion=1894, ficha=null}
  Nuevo t▯tulo (obligatorio):El nuevo libro de la selva
  Nuevo autor (obligatorio):Rudyard Kipling
  Nueva editorial (opcional):Macmillian Publishers
  Nuevo a▯o (opcional, ENTER para omitir):2004
  OK: Libro actualizado.

  ============================
     Sistema Libro▯Ficha (TFI)
  ============================
  1) Listar Libros activos
  2) Insertar Libro + Ficha (transacci▯n A?B)
  3) Actualizar Libro
  4) Eliminar Libro (baja l▯gica + su Ficha)
  5) Buscar Libro por ID
  6) Buscar Libro por ISBN  (JOIN con Ficha)
  0) Salir
  Elija una opci▯n: 1

  == Libros activos ==
  Libro{id=1, eliminado=false, titulo='Introducci▯n a la Programaci▯n', autor='Juan P▯rez', editorial='Editorial Alfa', anioEdicion=2018, ficha=null}
  Libro{id=5, eliminado=false, titulo='El nuevo libro de la selva', autor='Rudyard Kipling', editorial='Macmillian Publishers', anioEdicion=2004, ficha=null}
```

- Vemos su impacto luego de la actualización en el gestor de la base de datos (Título, año del Libro con ID=5)

En la columna “eliminado” destacamos un valor 0 para libros activos y un valor 1 para eliminados por baja lógica.



- Procedemos a simular una baja lógica en el mismo libro que creamos con ID=5

```
== Libros activos ==
Libro{id=1, eliminado=false, titulo='Introducci□n a la Programaci□n', autor='Juan P□rez', editorial='Editorial Alfa', anioEdicion=2018, ficha=null}
Libro{id=5, eliminado=false, titulo='El nuevo libro de la selva', autor='Rudyard Kipling', editorial='Macmillian Publishers', anioEdicion=2004, ficha=null}

==============================
   Sistema Libro□Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci□n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l□gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci□n: 5

== Buscar Libro por ID ==
ID: 5
```

Output

- Luego de encontrar el Id asignado (Opción 4 del menú), procedemos a eliminar aplicando la baja lógica del sistema.

```
== Buscar Libro por ID ==
ID: 5
Encontrado: Libro{id=5, eliminado=false, titulo='El nuevo libro de la selva', autor='Rudyard Kipling', editorial='Macmillian Publishers', anioEdicion=2004, ficha=null}

==============================
   Sistema Libro□Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci□n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l□gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci□n: 4

== Eliminar Libro (baja l□gica) ==
ID del libro a eliminar:5
```

- Baja realizada

```
==============================
   Sistema Libro□Ficha (TFI)
==============================
1) Listar Libros activos
2) Insertar Libro + Ficha (transacci□n A?B)
3) Actualizar Libro
4) Eliminar Libro (baja l□gica + su Ficha)
5) Buscar Libro por ID
6) Buscar Libro por ISBN  (JOIN con Ficha)
0) Salir
Elija una opci□n: 4

== Eliminar Libro (baja l□gica) ==
ID del libro a eliminar:5
OK: Libro y Ficha marcados como eliminados.
```

- Conclusiones y mejoras futuras

El trabajo permitió implementar exitosamente una aplicación Java con arquitectura en capas, aplicando el patrón DAO y transacciones con MySQL. Se logró modelar la relación 1→1 unidireccional entre Libro y FichaBibliográfica, cumpliendo con todos los requerimientos técnicos solicitados. La capa Service demostró ser efectiva para orquestar operaciones transaccionales y validaciones de negocio, garantizando la consistencia de datos.

En base a la experiencia del trabajo en equipo, habilidades adquiridas y consultas relacionadas en IA sobre nuestro trabajo llegamos a tener en cuenta estas futuras mejoras:

- Implementar interfaz gráfica con JavaFX o Swing para mejor experiencia de usuario.
- Agregar sistema de logs con archivos para registrar operaciones y errores.
- Desarrollar más validaciones de negocio (ej: email válido, rangos de precios) simulando un entorno de e-commerce.
- Implementar paginación para listados grandes de libros.
- Agregar búsquedas avanzadas por múltiples criterios.
- Desarrollar exportación de datos a formatos CSV o PDF.
- Implementar backup automático de la base de datos.
- Agregar manejo de imágenes de portadas de libros.
- Desarrollar sistema de copias de seguridad automáticas.
- Mejorar manejo de errores con mensajes más descriptivos.

- Anexo y recursos digitales

- Repositorio de Código Fuente:
Plataforma: GitHub
Enlace: https://github.com/manuG86/TPIntegrador_Grupo83
Descripción: Contiene todo el código fuente, documentación técnica y recursos del proyecto.

- Video Demostrativo:
Plataforma: YouTube
Enlace: https://youtu.be/zx_SPndM1wc
Descripción: Demostración en video del funcionamiento de la aplicación.