TRABAJO PRÁCTICO INTEGRADOR PROGRAMACIÓN II

Grupo Nº 65

Integrantes:

- Giorgio Buttigliero
- Juan Ignacio Ceballo
- Santino Hipólito Del Corro
- Agustín Nahuel Díaz Serafini

1. Integrantes y roles

- **Giorgio Buttigliero – Programador:**
  Fue responsable de implementar las clases principales del modelo. Se encargó de desarrollar la estructura base de las entidades, definir atributos, realizar validaciones básicas y colaborar en la carga de datos desde la interfaz por consola. Su participación permitió asegurar una correcta representación de los objetos del dominio.
- **Juan Ignacio Ceballo – Líder Técnico:**
  Coordinó el trabajo de los demás integrantes, organizó el repositorio, definió convenciones de codificación y revisó la estructura general del proyecto. También supervisó la arquitectura por capas, garantizando coherencia entre módulos y manteniendo el orden dentro de los paquetes del proyecto.
- **Santino Hipólito Del Corro – Programador / Capa Service:**
  Implementó la capa Service, que contiene la lógica de negocio. Su tarea fue fundamental para aplicar validaciones previas al acceso a la base de datos y para coordinar las operaciones entre el menú y los DAO. Además, contribuyó a mejorar el flujo de interacción del usuario.

- **Agustín Nahuel Díaz Serafini – DBA / DAO:**
  Diseñó las tablas en MySQL y elaboró el script de creación. Implementó las clases DAO responsables del CRUD, se ocupó del manejo de claves foráneas y realizó pruebas de consistencia e integridad. También ejecutó consultas SQL para verificar que la relación 1→1 se cumpliera correctamente.

2. Elección del dominio y justificación

El dominio elegido fue un **Sistema de Gestión de Libros y Fichas Bibliográficas**, un escenario que ofrece un equilibrio ideal entre simplicidad conceptual y desafío técnico.

Justificación del dominio

1. **Adecuado para relaciones 1→1**
   El caso de uso requiere que cada libro tenga una ficha bibliográfica única. Esto facilita implementar la relación 1→1 de manera limpia y concreta, cumpliendo con lo solicitado en el trabajo.
2. **Permite CRUD completo**
   Se pueden realizar altas, bajas, modificaciones y consultas tanto de libros como de fichas.
3. **Compatible con arquitectura por capas**
   El dominio se presta a dividir responsabilidades entre presentación, servicios, DAO y base de datos.
4. **Suficientemente flexible**
   A futuro, podrían agregarse entidades adicionales como autores, categorías, editoriales, préstamos o usuarios.
5. **Facilita validaciones reales**
   Por ejemplo:
   - Evitar ISBN duplicados
   - Verificar que un libro no tenga más de una ficha
   - Validar existencia de IDs antes de actualizar

6. **Representa un caso de estudio frecuente**
   La literatura es un dominio comúnmente utilizado para enseñar objetos, relaciones y persistencia.

3. Diseño: decisiones clave y UML

El diseño del sistema está basado en programación orientada a objetos, creando entidades que representan conceptos del dominio.

Entidades principales

Libro

Incluye datos esenciales de la obra:

- Identificador único
- Título
- Autor
- Año de publicación
- Editorial

FichaBibliografica

Contiene información complementaria:

- Identificador único
- ISBN
- Resumen
- Edición
- idLibro (clave foránea única)

Relación 1 a 1

La relación entre Libro y FichaBibliografica es estrictamente 1→1:

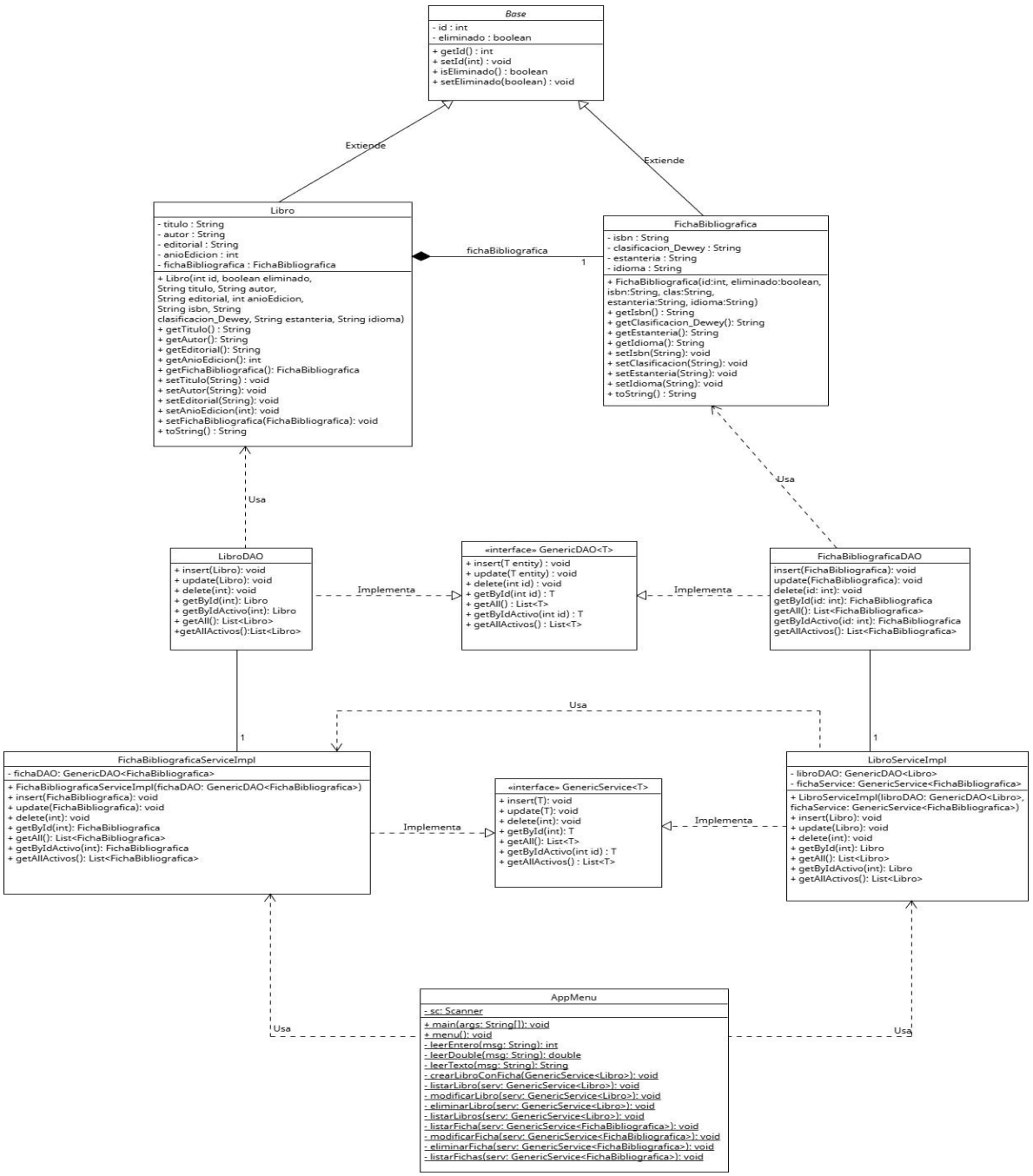
- Un libro → una única ficha

Una ficha → corresponde a un solo libro

Se decidió implementar esto con:

- Una FK UNIQUE en fichas_bibliograficas, apuntando a la PK de libros.

Ventaja:
Mantiene independencia entre tablas y respeta las buenas prácticas del diseño de bases de datos.



4. Arquitectura por capas

La arquitectura aplicada sigue un diseño clásico de múltiples capas donde cada módulo cumple una función específica.

Capa de Presentación

Ubicada en el paquete Main.
Incluye:

- Main.java
- AppMenu.java

Responsabilidades:

- Interactuar con el usuario.
- Solicitar entradas desde teclado.
- Mostrar resultados por consola.
- Enviar datos validados a la capa Service.

Capa Service

Implementa la lógica de negocio.
Sus funciones principales son:

- Validar campos obligatorios
- Evitar duplicados
- Verificar existencia de objetos antes de modificarlos
- Preparar los datos para los DAO

Capa DAO

Contiene las clases que interactúan directamente con la base de datos usando JDBC.

Incluye métodos:

- insert
- update
- delete

- getByIdActivos
- getAllActivos

Se utilizan PreparedStatement para evitar inyección SQL.

Capa Config

Maneja la conexión a MySQL con:

- DatabaseConnection
- DatabaseConnectionPool

Capa Base de Datos

Incluye el archivo creacionTablas.sql, donde se definen:

- Tablas
- Tipos de datos
- PK y FK
- Restricciones

5. Persistencia, estructura y transacciones

La base de datos se diseñó en MySQL, implementando dos tablas relacionadas:

Tabla libros

- id INT PK AUTO_INCREMENT
- titulo VARCHAR
- autor VARCHAR
- anio INT
- editorial VARCHAR

Tabla fichas_bibliograficas

- id INT PK
- idLibro INT FK UNIQUE

- isbn VARCHAR
- resumen TEXT

Flujo típico de persistencia

1. El usuario ingresa datos desde AppMenu.
2. Service valida:
    - Campos vacíos
    - ISBN
    - Regla 1→1
3. Service llama al DAO.
4. DAO prepara la sentencia SQL.
5. MySQL ejecuta la operación.
6. Se produce commit automático.

En caso de error:

- Se captura la excepción
- Se informa al usuario
- Se aborta la operación

6. Validaciones y reglas de negocio

Validaciones implementadas:

- Campos obligatorios: título, autor, año y editorial.
- ISBN con formato correcto.
- idLibro debe existir antes de crear su ficha.
- No se permite eliminar un libro con una ficha asociada sin antes eliminar la ficha.
- No se permiten fichas duplicadas para un mismo libro.

Reglas de negocio principales

- Un libro puede existir sin ficha, pero una ficha **no puede existir sin libro**.
- Un libro no puede tener más de una ficha (FK UNIQUE).
- Solo se pueden modificar o eliminar libros/fichas existentes.

7. Pruebas realizadas

Se realizaron pruebas exhaustivas sobre los módulos implementados.

Pruebas de alta

- Agregar libro → correcto
- Agregar ficha bibliográfica → chequea existencia del libro

Pruebas de baja

- Eliminar libro sin ficha → OK
- Eliminar libro con ficha → error controlado
- Eliminar ficha → OK

Pruebas de modificación

- Cambiar título o autor
- Reemplazar ISBN de ficha
- Error si el ID no existe

Pruebas de consultas

- Listado completo
- Búsqueda por ID
- Consulta combinada con JOIN

8. Conclusiones y mejoras futuras

Conclusiones

- Se logró implementar correctamente un sistema que cumple con todos los requisitos del TPI.
- La arquitectura por capas facilita el mantenimiento y la expansión del sistema.
- La relación 1→1 fue aplicada de manera correcta con FK UNIQUE.
- El patrón DAO permitió una clara separación entre lógica y persistencia.
- La base de datos mantiene integridad y consistencia.

Mejoras futuras

- Migrar a Hibernate o JPA para evitar SQL manual.
- Agregar interfaz gráfica o aplicación web.
- Añadir más entidades (Autores, Editoriales, Usuarios).
- Aplicar transacciones manuales para operaciones complejas.
- Integrar una API REST para exponer los datos.

6. Fuentes y herramientas utilizadas

- Documentación oficial de Java SE
- Documentación de MySQL
- NetBeans IDE 17
- Git y GitHub
- Material de la cátedra UTN – Programación II