TRABAJO FINAL INTEGRADOR

PROGRAMACION II

Alumnas:

- Demarchi, Eugenia
- Gonzalez, Erika Samantha
- Ortellado, Karen
- Rege, María Soledad

Tecnicatura Universitaria en Programación
Universidad Tecnológica Nacional
Año 2025

Índice

Trabajo Final Integrador - Programación II

Introducción

Este Trabajo Integrador Final tiene como finalidad aplicar y consolidar los contenidos estudiados durante la cursada de la asignatura Programación 2, a través de la creación de un proyecto completo que articula diseño orientado a objetos, arquitectura por capas, manejo de excepciones, control de transacciones y acceso a datos mediante JDBC.

De esta manera, se implementa un sistema funcional basado en un dominio previamente modelado, separando correctamente responsabilidades entre capas, construyendo servicios con lógica de negocio realista, usando adecuadamente patrones como DAO y Service y validando los datos antes de su persistencia.

Para complementar y fortalecer el aprendizaje, se utilizó de manera responsable herramientas de inteligencia artificial como apoyo conceptual. Su uso se limitó a la validación de ideas, consultas aisladas y verificación de buenas prácticas, manteniendo siempre la autonomía en el diseño, la implementación y la toma de decisiones técnicas.

Objetivos

El objetivo de este trabajo consiste en desarrollar una aplicación en Java que modele dos clases relacionadas mediante una asociación unidireccional 1 a 1, persistiendo datos en una base de datos relacional mediante JDBC y patrón DAO, con operaciones transaccionales y un menú de consola para controlar el CRUD.

Roles de cada estudiante

1. ***Demarchi, Eugenia:*** Documentación, diseño del UML, desarrollo de entidades, desarrollo de la *AppMenú*.
2. ***Gonzalez, Erika Samantha:*** Documentación, desarrollo de la capa *DAO* y transacciones.
3. ***Ortellado, Karen:*** Documentación, desarrollo de la capa de *Config* con su conexión a la base de datos, desarrollo de scripts SQL.
4. ***Rege, María Soledad:*** Documentación, desarrollo de la capa *Services*, reglas de negocio y validaciones.

Elección del dominio

Para este proyecto se trabajó con las entidades **Libros y Fichas Bibliográficas**, elegidas a partir de los modelos propuestos. El vínculo unilateral entre ambas reproduce una situación frecuente en sistemas de gestión bibliotecaria, donde cada libro debe contar con una ficha que consolide información estructurada.

A su vez, este dominio es lo suficientemente simple para centrarse en la implementación de la arquitectura por capas, pero también flexible para escalar hacia sistemas más complejos (préstamos, lectores, autores o géneros), lo que lo convierte en un punto de partida para futuros desarrollos.

Diseño

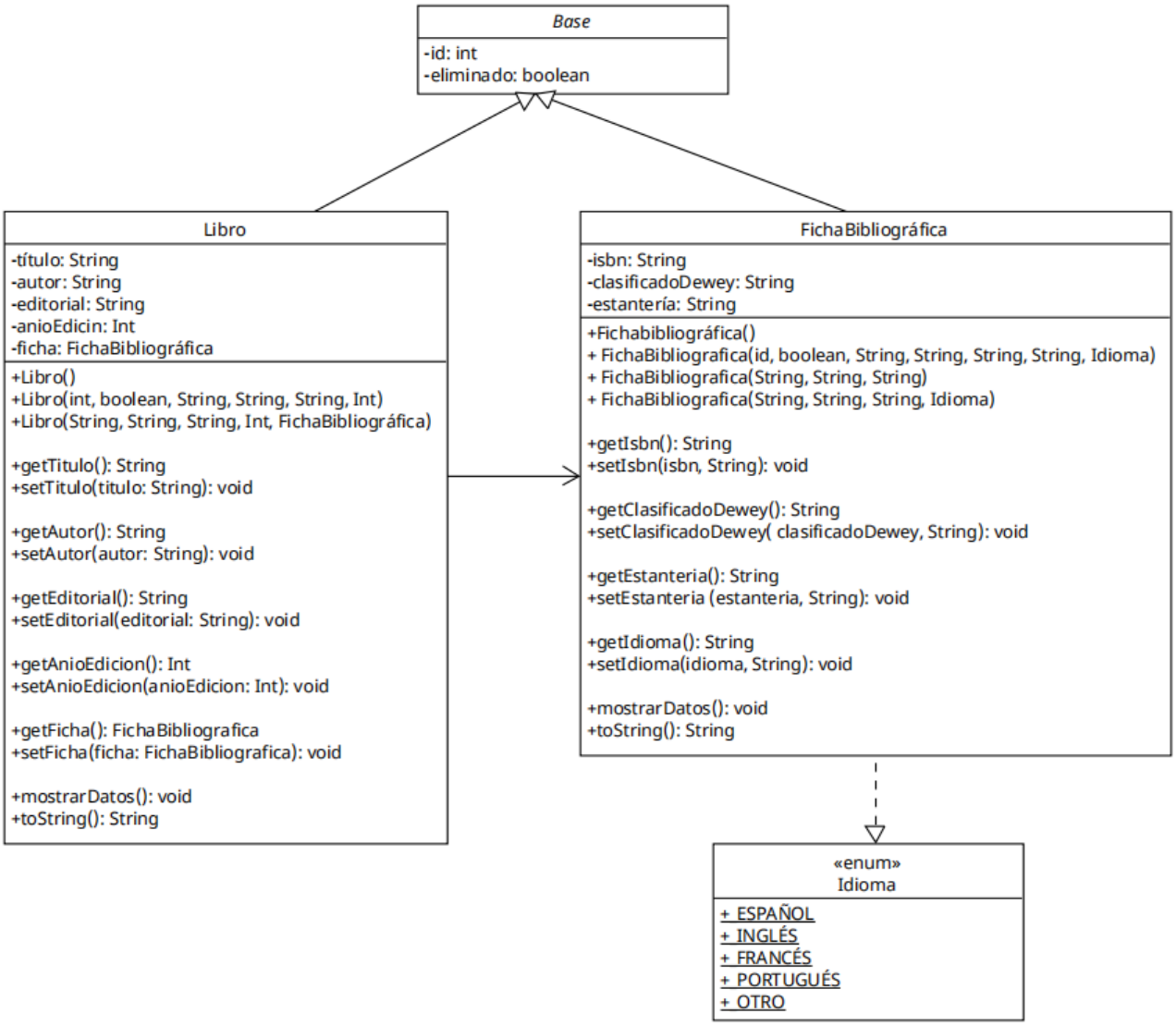
Partiendo del CRUD seleccionado: Clase Libro y Clase FichaBibliográfica, se procede a modelar las mismas. La relación deberá ser de tipo 1 a 1: cada Libro tiene una sola FichaBibliográfica asociada y cada FichaBibliográfica pertenece a un solo Libro. Desde ésta perspectiva, podemos definir las entidades y sus correspondientes atributos, ambas heredando la clase abstracta Base:

BASE		
Atributo	**Tipo**	**Descripción**
id	Int (Primary Key)	Identificador único
eliminado	Boolean	Indica si el registro fue eliminado lógicamente

FICHA BIBLIOGRAFICA		
Atributo	**Tipo**	**Descripción**
id	Int (Primary Key)	Identificador único de la ficha bibliográfica
eliminado	Boolean	Indica si el registro fue eliminado lógicamente
isbn	String	Código ISBN del libro (identificador internacional)
clasificacionDewey	String	Clasificación según el sistema Dewey
estantería	String	Ubicación física del libro
idioma	Idioma (Enum)	Idioma del libro (*Español, Inglés, Francés, Portugués, Otro)*

LIBRO		
Atributo	**Tipo**	**Descripción**
id	Int (Primary Key)	Identificador único del libro
eliminado	Boolean	Indica si el libro fue eliminado lógicamente
titulo	String	Título del libro
autor	String	Autor o autores
editorial	String	Editorial del libro
anioEdicion	Int	Año de edición del libro
ficha	FichaBibliografica (Foreign Key, asociación unidireccional)	Identificador de la ficha bibliográfica asociada

Diagrama UML



Arquitectura por capas

El proyecto se encuentra dividido en diferentes capas que garantizan una organización clara y una responsabilidad bien definida para cada componente. De esta manera, el código se mantiene mantenible y escalable a futuro. A continuación se describen las tareas que contiene cada una.

Capa Model

La capa Model contiene las entidades Libro y FichaBibliografica, con sus atributos, constructores, getters, setters y métodos complementarios. También incluye la clase abstracta Base, que centraliza los atributos *id* y *eliminado*, reutilizados por todas las entidades del dominio. Su función principal es representar de manera fiel la estructura de los datos y las relaciones del sistema, incorporando únicamente la lógica mínima necesaria dentro de cada clase (como métodos de visualización o pequeñas validaciones internas). Las clases del modelo se mantienen completamente independientes de la lógica de negocio y de la persistencia, permitiendo que puedan ser utilizadas por cualquier capa

sin generar dependencias innecesarias. Esto garantiza un diseño limpio, reutilizable y alineado con los principios de la programación orientada a objetos.

Capa DB

La capa DB comprende tanto la estructura de la base de datos, diseñada en MySQL, como la clase desarrollada en Java que permite al sistema conectarse y operar con ella. Esta capa constituye un componente fundamental para el correcto desarrollo del trabajo práctico, dado que constituye la base sobre la cual se construyen las capas posteriores (DAO, Service y Menú), que interactúan entre sí para ejecutar las operaciones CRUD del proyecto.

En primer lugar, se realizó el diseño general tal como se describió anteriormente: Libro se definió como la entidad principal, mientras que FichaBibliográfica constituye una entidad dependiente de Libro. Para representar esta relación se utilizó una clave foránea en la tabla *ficha_bibliografica*.
Asimismo, se incorporaron los campos:

- eliminado, para implementar la baja lógica;
- created_at y updated_at, como marcas temporales automáticas;
- además de índices destinados a optimizar las búsquedas.

Todos estos elementos se encuentran plasmados en el archivo **create_db.sql**, el cual permite generar la estructura completa de la base de datos desde cero. Este archivo está preparado para que cualquier usuario pueda recrear fácilmente el esquema en su propio entorno. Incluye la creación de la base de datos, la definición de ambas tablas con sus claves primarias, la clave foránea con restricción UNIQUE, los índices adicionales y la configuración de codificación *utf8mb4*.

Posteriormente, en el archivo **seed_data.sql** se elaboró un script destinado a cargar datos de prueba en las tablas. En este caso, se incluye la carga inicial de tres libros y tres fichas bibliográficas. Ambos archivos SQL se encuentran organizados dentro de la carpeta sql.

Por otro lado, se implementó la clase **DatabaseConnection.java**, responsable de establecer la conexión entre el sistema y la base de datos. Esta clase, ubicada en el paquete *trabajo_integrador.config*, se encarga de cargar el driver JDBC de MySQL, validar la URL, el usuario y la contraseña, y ofrecer el método estático getConnection(), que centraliza la obtención de conexiones. De este modo, se garantiza que el resto del proyecto no dependa de detalles específicos de configuración y que cualquier DAO pueda acceder a la base de datos mediante la instrucción:

```
Connection conn = DatabaseConnection.getConnection();
```

Finalmente, se desarrolló la clase **TestConnection.java** con el propósito de verificar el correcto funcionamiento del mecanismo de conexión. Este archivo permite corroborar, mediante la salida por

consola, si la conexión se establece satisfactoriamente. Su uso facilita la validación tanto de la capa DAO como del sistema completo.

Capa DAO

La capa DAO constituye la capa de persistencia del sistema. Su única responsabilidad es abstraer la lógica de comunicación directa con la base de datos, proveyendo a la capa de Servicio una interfaz clara para interactuar con las entidades del modelo.

Las responsabilidades se centran en ser la única capa que contiene y ejecuta sentencias SQL. Utiliza PreparedStatement de forma sistemática para prevenir ataques de inyección SQL y manejar los parámetros de manera eficiente.

La arquitectura de esta capa se centra en la interfaz GenericDAO<T>, que define el contrato estándar para las operaciones CRUD para cualquier entidad. Además, incluye un método recuperar para manejar la lógica de borrado lógico.

Para dar soporte a la capa de Servicio, la capa DAO hace uso del polimorfismo para ofrecer un patrón dual en el manejo de conexiones para cada operación:

Métodos sin parámetro Connection: Estos métodos gestionan su propio ciclo de vida de conexión. Abren una nueva conexión y la cierran automáticamente al finalizar.

Métodos con parámetro Connection: Estos métodos utilizan la conexión que reciben como parámetro y no la cierran. Permitiendo a la capa de Servicio orquestar operaciones complejas dentro de una única transacción, manejando manualmente el commit() o rollback() sobre esa conexión.

Por último, otra implementación importante que se realiza es el del borrado lógico. En lugar de ejecutar un DELETE físico de la base de datos, el método eliminar() realiza una operación UPDATE que cambia la columna eliminado de FALSE a TRUE. Consecuentemente, las demás operaciones están diseñadas para incluir explícitamente la cláusula WHERE eliminado = FALSE en sus consultas. Garantizando que sólo se opere sobre registros activos. Para completar este patrón, también se provee un método recuperar() que revierte el borrado, volviendo a establecer la columna eliminado = FALSE para el registro especificado.

Capa Services

La capa Service es responsable de implementar la lógica de negocio del sistema y de aplicar las validaciones necesarias antes de interactuar con la capa de persistencia. Lo servicios se apoyan en la capa DAO para las operaciones básicas sobre la base de datos, es decir, no maneja transacciones de forma general, salvo en un caso particular: la operación de inserción que combina la creación de una ficha bibliográfica y de un libro dentro de una misma transacción.

Para este caso puntual, se requiere abrir manualmente una conexión y deshabilitar el auto-commit mediante setAutoCommit(false). Así, ambas operaciones se ejecutan de manera atómica, garantizando la coherencia de los datos. Si todo finaliza correctamente, se realiza un commit(), mientras que ante cualquier error se ejecuta un rollback() para revertir los cambios.

En cuanto a las validaciones de negocio, la capa Service se encarga de asegurar:

- Que los campos obligatorios no estén vacíos.
- Que los textos cumplan con los límites máximos de longitud definidos.
- Que los valores numéricos sean correctos y dentro de los rangos permitidos.
- Que la entidad exista previamente antes de actualizarla o eliminarla.
- Que una ficha bibliográfica no esté asociada a otro libro, respetando la relación 1 a 1.
- Que los valores del dominio (como el idioma) correspondan al tipo definido por el sistema.

De esta manera, la capa Service actúa como intermediaria entre la interfaz de usuario y la capa DAO, encapsulando la lógica de negocio y garantizando la consistencia del sistema.

Capa Menú

La capa Main es la responsable de gestionar la interacción con el usuario. Desde esta capa se pueden ejecutar las operaciones básicas del CRUD para cada entidad. Esta capa se organiza en cuatro clases principales:

AppMenu funciona como el orquestador principal de la aplicación. Su responsabilidad es administrar el ciclo de vida del menú y coordinar las interacciones entre los distintos componentes. Actúa como ensamblador central, creando e inyectando las dependencias necesarias (DAOs → Services → MenuHandler), sin involucrarse en lógica de negocio.

Responsabilidades puntuales:

- Crear y administrar un único objeto Scanner, evitando múltiples lecturas sobre System.in.
- Inicializar y ensamblar todas las dependencias (DAOs, Services y MenuHandler).
- Ejecutar el bucle principal del menú mediante el método run().
- Interpretar la opción ingresada por el usuario y delegar la acción correspondiente al MenuHandler.
- Cerrar recursos al finalizar la ejecución (como el Scanner).

MenuDisplay se encarga de mostrar el menú de la aplicación. Contiene únicamente métodos estáticos destinados a la visualización y no mantiene estado.

MenuHandler funciona como el controlador de las operaciones del menú. Administra toda la lógica de interacción con el usuario para ejecutar las operaciones CRUD. No contiene lógica de negocio: las

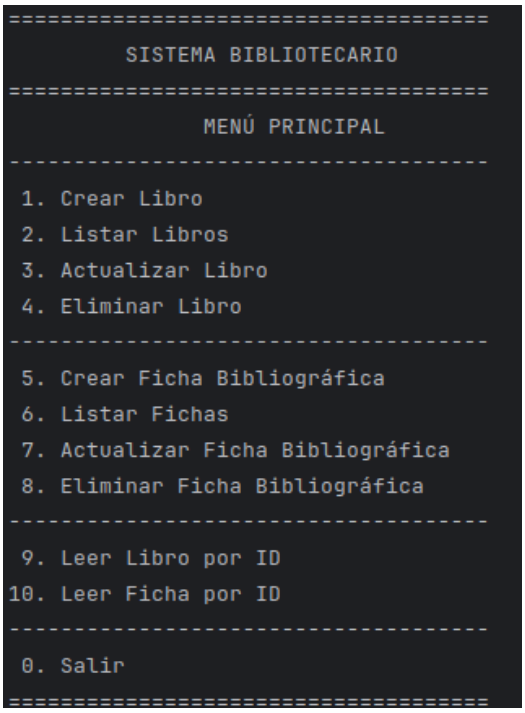
validaciones propias del dominio se realizan en la capa Service, que en caso de error lanza excepciones capturadas por el MenuHandler para informar al usuario.

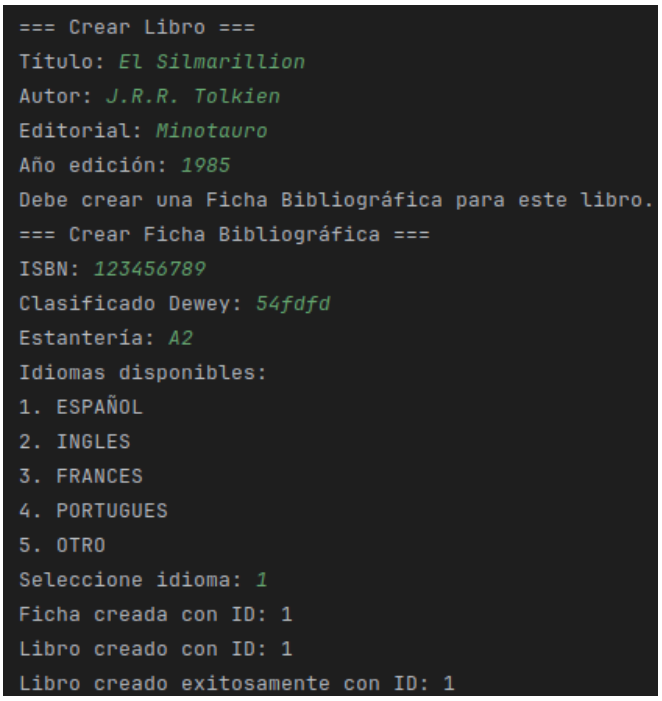
Sin embargo, MenuHandler sí realiza validaciones de entrada (números válidos, opciones del menú, rangos básicos, campos vacíos, confirmaciones), necesarias para asegurar que la interacción con el usuario sea correcta y evitar errores de tipeo.

Principal actúa como un punto de entrada alternativo de la aplicación. Es una clase simple que delega inmediatamente el control a *AppMenu*.

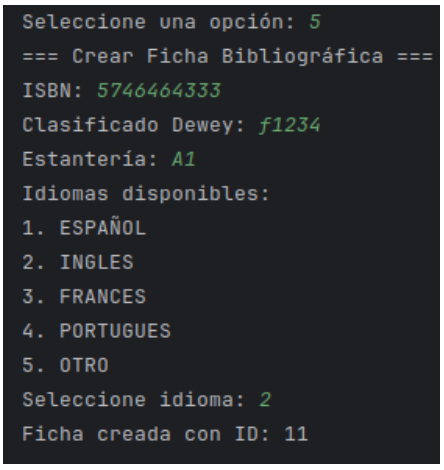
Pruebas realizadas

Creación de Libro + Ficha





Crear Ficha



Listar

```
=== Listar Libros ===
ID: 1, Titulo: El Silmarillion, Autor: J.R.R. Tolkien, Editorial: Minotauro
   Ficha ID: 1 | ISBN: 123456789
ID: 2, Titulo: El Señor de los Anillos, Autor: J.R.R. Tolkien, Editorial: Minotauro
   Ficha ID: 2 | ISBN: 321654987
```

```
=== Listar Fichas Bibliográficas ===
ID: 2 | ISBN: 321654987 | Dewey: 456fgfdgd | Estantería: A2 | Idioma: ESPAÑOL
ID: 4 | ISBN: 123645498 | Dewey: fsdfs | Estantería: fdsfs | Idioma: ESPAÑOL
ID: 5 | ISBN: 4567489746 | Dewey: DASDAS | Estantería: DFSD | Idioma: ESPAÑOL
ID: 8 | ISBN: 987654321 | Dewey: asdasd | Estantería: sdas | Idioma: ESPAÑOL
ID: 10 | ISBN: 258963147 | Dewey: asdas55 | Estantería: A4 | Idioma: ESPAÑOL
ID: 11 | ISBN: 5746464333 | Dewey: f1234 | Estantería: A1 | Idioma: INGLÉS
```

Actualizar

```
=== Actualizar Libro ===
ID del libro a actualizar: 7
Nuevo título (actual: La Comunidad del Anillo): La Comunidad del Anillo
Nuevo autor (actual: J.R.R. Tolkien): J.R.R. Tolkien
Nueva editorial (actual: Minotauro): Minotauro
Nuevo año (actual: 1953): 1955
¿Desea reemplazar la ficha bibliográfica? (s/n):
n
Libro actualizado exitosamente.
```

```
=== Actualizar Ficha Bibliográfica ===
Ingrese el ID de la ficha a actualizar: 3
Ficha encontrada:
=== Ficha Bibliográfica ===
Id: 3
ISBN: 213654
Clasificación Dewey: 54rter
Estantería: A4
Idioma: FRANCÉS
Eliminado: false
Nuevo ISBN (213654): 213654
Nuevo Clasificado Dewey (54rter): 54gfg
Nueva Estantería (A4): A4
Seleccione nuevo idioma (ENTER para mantener):
1. ESPAÑOL
2. INGLÉS
3. FRANCÉS
4. PORTUGUÉS
5. OTRO
Opción: 3
Ficha actualizada con éxito.
```

Eliminar

```
=== Eliminar Ficha Bibliográfica ===
Ingrese el ID de la ficha a eliminar: 6
Ficha encontrada:
=== Ficha Bibliográfica ===
Id: 6
ISBN: asdasda
Clasificación Dewey: ssdasa
Estantería: asasd
Idioma: ESPAÑOL
Eliminado: false
¿Está seguro que desea eliminarla? (S/N): s
Ficha eliminada correctamente.
```

```
=== Eliminar Libro ===
ID del libro a eliminar: 4
¿Está seguro que desea eliminarla? (S/N): s
Libro eliminado exitosamente.
```

Buscar por ID

```
=== Buscar Libro por ID ===
Ingrese el ID: 1
--- Libro Encontrado ---
ID: 1
Título: El Silmarillion
Autor: J.R.R. Tolkien
Editorial: Minotauro
Año: 1985
Ficha Bibliográfica Asociada:
 - Ficha ID: 1
 - ISBN: 123456789
 - Dewey: 54fdfd
 - Estantería: A2
 - Idioma: ESPAÑOL
```

```
=== Buscar Ficha Bibliográfica por ID ===
Ingrese el ID: 5
--- Ficha Encontrada ---
ID: 5
ISBN: 4567489746
Clasificado Dewey: DASDAS
Estantería: DFSD
Idioma: ESPAÑOL
```

Conclusiones

La realización de este Trabajo Final Integrador ha permitido aplicar y consolidar de manera exitosa los conceptos fundamentales de la asignatura Programación II mediante la creación de una aplicación Java funcional.

El pilar del proyecto fue la implementación de una arquitectura por capas, logrando una separación de responsabilidades clara y efectiva: el Modelo representa los datos, las capas DAO y DB abstrajeron la persistencia, Servicios encapsuló la lógica de negocio y el Menú gestionó la interacción con el usuario.

Técnicamente, destacamos la correcta gestión de transacciones para garantizar la atomicidad en operaciones complejas y la implementación de borrado lógico en el DAO para la integridad y recuperación de datos. El resultado es un sistema robusto y mantenible que cumple los objetivos del CRUD y valida el aprendizaje de JDBC y el diseño orientado a objetos.