README - Proyecto: Biblioteca (DAO Pattern)

Resumen generado automáticamente del código fuente proporcionado en 'TrabajoDAOFin.txt'. El proyecto implementa un ejemplo de gestión de una biblioteca usando el patrón DAO (Data Access Object) con capas separadas para modelos, DAOs, implementación de DAOs, servicio y una interfaz de consola (Main).

Estructura general y paquetes detectados

- dao
- dao.autor
- dao.libro
- dao.libro_autor
- dao.prestamo
- dao.usuario
- model
- org.example
- service

Clases e interfaces principales detectadas

- Autor
- AutorDAO
- AutorImpl
- BibliotecaService
- ConexionBD
- Libro
- LibroAutor
- LibroAutorDAO
- LibroAutorImpl
- LibroDAO
- LibroImpl
- Main
- Prestamo
- PrestamoDAO
- PrestamoImpl
- Usuario
- UsuarioDAO
- UsuarioImpl

Modelos (package model)

- Autor
- Libro
- LibroAutor
- Prestamo
- Usuario

DAOs (interfaces e implementaciones)

El proyecto define interfaces DAO para Autor, Libro, Usuario, Prestamo y la relación LibroAutor, junto con sus implementaciones que interactúan con una base de datos MySQL (conector JDBC).

Interfaces DAO detectadas:

- AutorDAO
- LibroAutorDAO
- LibroDAO
- PrestamoDAO
- UsuarioDAO

Implementaciones DAO detectadas:

- AutorImpl
- LibroAutorImpl
- LibroImpl
- PrestamoImpl
- UsuarioImpl

Capa de servicio: BibliotecaService

La clase BibliotecaService orquesta operaciones de alto nivel: registrar autores/libros/usuarios/préstamos, listar y eliminar, así como operaciones compuestas como registrar un libro junto con su autor (crea la relación en la tabla intermedia). La capa de servicio captura excepciones y realiza mensajes por consola cuando algo falla.

Interfaz de consola (Main)

El archivo 'Main' provee un menú en consola (Scanner) con submenús para Autor, Libro, Préstamo y Usuario. Permite operaciones CRUD básicas y llama a BibliotecaService para ejecutar las acciones contra la base de datos. Formato de entrada para fechas en préstamos: 'YYYY-MM-DD'.

Conexión a base de datos (ConexionBD)

La configuración JDBC está en ConexionBD con URL 'jdbc:mysql://localhost:3306/biblioteca' y credenciales por defecto 'root' / '123456789'. La clase mantiene una Connection estática. Hay un constructor publico que asigna USER/PASSWORD y llama a conectar().

Observaciones automáticas y recomendaciones

- La clase ConexionBD define un atributo estático 'conexion' y un método estático getConexion(), pero la inicialización de la conexión ocurre en el constructor ConexionBD(...). Si el resto del código llama a ConexionBD.getConexion() sin crear una instancia de ConexionBD, 'conexion' puede ser null y provocar NullPointerException al usarla. Se recomienda proporcionar un método estático de inicialización o mantener la conexión gestionada internamente (p.ej. singleton).
- Se detectan nombres de tabla/columnas inconsistentes relacionados con la relación libro-autor (p. ej. 'libroAutor' vs 'libro_autor'). Esto puede provocar SQLExceptions en tiempo de ejecución si la base de datos no tiene exactamente esas tablas/columnas. Asegúrese de unificar la convención de nombres entre todas las clases DAO y el esquema de la base de datos.
- En LibroImpl.updateLibro se actualiza únicamente el campo 'titulo' pero no 'isbn'. Si la intención es permitir actualizar ISBN también, hay que incluirlo en la sentencia UPDATE y en los parámetros.
- PrestamoImpl realiza LocalDate.parse(...) sobre las fechas recibidas; si la fecha no cumple el formato ISO 'YYYY-MM-DD' lanzará DateTimeParseException. Asegúrese de validar/normalizar el formato antes de llamar al servicio.
- Uso mayoritariamente correcto de try-with-resources en los DAO para asegurar el cierre de conexiones, sentencias y resultados.

Fragmentos relevantes detectados (extractos)

```
public class AutorImpl implements AutorDAO { @Override public void
addAutor(Autor autor) { String sql =
```

```
"INSERT INTO autor (nombre) VALUES (?)"; try ( Connection conn =
ConexionBD.getConexion();
PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS) ) {
ps.setString(1, autor.getNombre()); ps.executeUpdate(); try (ResultSet rs =
ps.getGeneratedKeys()) { if (rs.next()) autor.setId(rs.getInt(1)); } } catch
(SQLException ex) { throw new RuntimeException(ex); } System.out.println("DAO:
Autor
insertado ->
```

```
public class LibroImpl implements LibroDAO { @Override public void
addLibro(Libro libro) throws SQLException {
String sql = "INSERT INTO libro (titulo, isbn) VALUES (?, ?)"; try (
Connection conn =
ConexionBD.getConexion(); PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS) ) { ps.setString(1, libro.getTitulo());
ps.setString(2,
libro.getIsbn()); ps.executeUpdate(); try (ResultSet rs =
ps.getGeneratedKeys()) {
if (rs.next()) libro.setId(rs.getInt(1)); } System.out.println("DAO: Libro
insertado -> " +
libro);
```

```
class ConexionBD { private static final String URL =
"jdbc:mysql://localhost:3306/biblioteca"; private
static String USER = "root"; private static String PASSWORD = "123456789";
private static Connection
conexion; public ConexionBD(String usuario, String contra) { USER = usuario;
PASSWORD = contra; try { conectar();
```

Archivo fuente analizado

/mnt/data/TrabajoDAOFin.txt

Cómo ejecutar / Notas finales

1) Crear la base de datos MySQL 'biblioteca' y las tablas necesarias: autor, libro, libro_autor (o libroAutor si usa esa convención), usuario y prestamo con columnas coherentes con las consultas SQL presentes en el código. 2) Ajustar credenciales en ConexionBD o instanciar ConexionBD con las credenciales deseadas antes de ejecutar la aplicación. 3) Compilar y ejecutar Main; la aplicación usa la consola para entradas. 4) Revisar y unificar nombres de tablas/columnas y la inicialización de la conexión para evitar errores en tiempo de ejecución.