MEMORIA TÉCNICA:
GESTIÓN DE BIBLIOTECA CON HIBERNATE

Módulo: Acceso a Datos (U3. Actividad Práctica)

ÍNDICE

Apartado 1: Preparación del Entorno

1.1 Explicación del Funcionamiento

En este apartado configuré el entorno de desarrollo, lo que incluyó la creación de la base de datos biblioteca_hibernate en MySQL, la configuración del proyecto Maven con las dependencias de Hibernate, MySQL Connector y Javax Persistence API (JPA). Seguidamente, la creación del archivo hibernate.cfg.xml apuntando a la BBDD, y la implementación de la clase HibernateUtil para gestionar el SessionFactory.

1.2 Fragmentos de Código Relevantes

```xml
<dependencies>
    <!--  Hibernate  -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.6.15.Final</version>
    </dependency>
    <!--  MySQL Connector  -->
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.3.0</version>
    </dependency>
    <!--  Javax Persistence API  -->
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
</dependencies>
```

Estructura del pom.xml (Dependencias)

```xml
<hibernate-configuration>
    <session-factory>
        <!--  Configuración de conexión a la base de datos  -->
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/biblioteca_hibernate</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">root</property>

        <!--  Dialecto de MySQL  -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <!--  Mostrar SQL en consola  -->
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>

        <!--  Gestión automática del esquema  -->
        <property name="hibernate.hbm2ddl.auto">validate</property>

        <!--  Pool de conexiones  -->
        <property name="hibernate.pool_size">10</property>

        <!--  Mapeo de entidades  -->
        <mapping class="com.dam.modelo.Autor"/>
        <mapping class="com.dam.modelo.Libro"/>
        <mapping class="com.dam.modelo.Ejemplar"/>
    </session-factory>
</hibernate-configuration>
```

Estructura del hibernate.cfg.xml

```java
package com.dam.util;

// Importa las librerías necesarias
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

/**
 *
 * @author byAlexLR
 */
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    // Bloque de inicialización
    static {
        try {
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Session Factory creation failed: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
```

Clase HibernateUtil.java

Apartado 2: Implementación de entidades

2.1 Explicación del Funcionamiento

Creé las tres clases de entidad (Autor, Libro, Ejemplar) con anotaciones JPA, mapeando la base de datos. Configuré las relaciones (OneToMany, ManyToOne) con cargas LAZY y tipos de cascada (ALL, orphanRemoval=true), incluyendo métodos "helper" para mantener la consistencia bidireccional.

2.2 Estructura de la Base de datos

TABLA AUTOR →

id_autor	nombre	apellidos	nacionalidad	fecha_nacimiento
1	Gabriel	García Márquez	Colombiana	1927-03-06
2	Isabel	Allende	Chilena	1942-08-02
3	Miguel	de Cervantes	Española	1547-09-29
NULL	NULL	NULL	NULL	NULL

id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
3	EJ-003-2024	PRESTADO	Estantería A-13	2
4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
NULL	NULL	NULL	NULL	NULL

← TABLA EJEMPLAR

TABLA LIBRO

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
2	El amor en los tiempos del cólera	978-84-397-0428-7	1985-09-05	464	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	3
NULL	NULL	NULL	NULL	NULL	NULL

2.3 Fragmentos de Código Relevantes

A continuación, se detalla la estructura de mapeo para las clases Autor, Ejemplar y Libro. Cabe destacar que las clases Autor y Libro incluyen métodos Helper diseñados para facilitar la adición y eliminación de datos.

```java
@Entity
@Table(name = "autores")
public class Autor {

    // Declaración de variables con Mapeo
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_autor")
    private Integer idAutor;

    @Column(name = "nombre")
    private String nombre;

    @Column(name = "apellidos")
    private String apellidos;

    @Column(name = "nacionalidad")
    private String nacionalidad;

    @Column(name = "fecha_nacimiento")
    private LocalDate fechaNacimiento;

    // Relación uno a muchos. Todas las operaciones se propagan a los libros y a los huérfanos.
    @OneToMany(mappedBy = "autor", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Libro> libros = new ArrayList<>();

    // Métodos Helper (Añade o elimina un libro al autor)
    public void addLibro(Libro libro) {
        libros.add(libro);
        libro.setAutor(this);
    }

    public void removeLibro(Libro libro) {
        libros.remove(libro);
        libro.setAutor(null);
    }
```

Entidad/Clase Autor.java

```java
@Entity
@Table(name = "ejemplares")
public class Ejemplar {

    // ENUM que define los posibles estados del Ejemplar
    public enum EstadoEjemplar {
        DISPONIBLE,
        PRESTADO,
        REPARACION,
        BAJA
    }

    // Declaración de variables con Mapeo
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_ejemplar")
    private Integer idEjemplar;

    @Column(name = "codigo_ejemplar")
    private String codigoEjemplar;

    @Enumerated(EnumType.STRING)
    @Column(name = "estado")
    private EstadoEjemplar estado;

    @Column(name = "ubicacion")
    private String ubicacion;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "id_libro")
    private Libro libro;
```

Entidad/Clase Ejemplar.java

```java
@Entity
@Table(name = "libros")
public class Libro {

    // Declaración de variables con Mapeo
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_libro")
    private Integer idLibro;

    @Column(name = "titulo")
    private String titulo;

    @Column(name = "isbn")
    private String isbn;

    @Column(name = "fecha_publicacion")
    private LocalDate fechaPublicacion;

    @Column(name = "numero_paginas")
    private int numeroPaginas;

    // Relación muchos a uno. Carga el autor.
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "id_autor")
    private Autor autor;

    // Relación uno a muchos. Todas las operaciones se propagan a los ejemplares y a los huérfanos.
    @OneToMany(mappedBy = "libro", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Ejemplar> ejemplares = new ArrayList<>();
```

```java
// Métodos Helper (Añade o elimina un ejemplar al libro)
public void addEjemplar(Ejemplar ejemplar) {
    ejemplares.add(ejemplar);
    ejemplar.setLibro(this);
}

public void removeEjemplar(Ejemplar ejemplar) {
    ejemplares.remove(ejemplar);
    ejemplar.setLibro(null);
}
```

Entidad/Clase Libro.java

Apartado 3: Operaciones Create

3.1 Explicación del Funcionamiento

Implementé los métodos para insertar nuevos datos en la BBDD. El método crearAutorConLibros, ya que con una sola llamada a session.persist(autor), Hibernate se encarga de insertar el autor. Luego sus dos libros, y finalmente los tres ejemplares asociados. El método agregarEjemplarALibroExistente muestra cómo buscar un libro (por ISBN) y añadirle un nuevo ejemplar.

3.2 Fragmentos de Código Relevantes con Resultado

A continuación, se detalla la estructura de creación de un autor y búsqueda de un libro para añadirle un nuevo ejemplar.

```java
// Diferentes métodos CRUD (CREATE, READ, UPDATE, DELETE).
public void crearAutorConLibros() {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Crea el Autor
        Autor autor = new Autor("Javier", "Marías", "Española", LocalDate.of(1951, 9, 20));

        // Crea el primer libro
        Libro libro1 = new Libro("Corazón tan blanco", "978-84-204-2954-3", LocalDate.of(1992, 3, 1), 368);

        // Añade ejemplares al libro
        Ejemplar ej1 = new Ejemplar("EJ-009-2024", EstadoEjemplar.DISPONIBLE, "Estantería E-10");
        Ejemplar ej2 = new Ejemplar("EJ-010-2024", EstadoEjemplar.DISPONIBLE, "Estantería E-10");
        libro1.addEjemplar(ej1);
        libro1.addEjemplar(ej2);

        // Crea el segundo libro
        Libro libro2 = new Libro("Mañana en la batalla piensa en mí", "978-84-204-2955-0", LocalDate.of(1994, 5, 5), 464);

        // Añade un ejemplar al libro
        Ejemplar ej3 = new Ejemplar("EJ-011-2024", EstadoEjemplar.DISPONIBLE, "Estantería E-11");
        libro2.addEjemplar(ej3);

        // Añade los libros al autor
        autor.addLibro(libro1);
        autor.addLibro(libro2);

        // Guarda el autor
        session.persist(autor);

        // Hace los cambios permanentes y visibles
        tx.commit();
        System.out.println("El autor ha sido creado exitosamente con " + autor.getLibros().size() + " libros.");
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL CREAR EL AUTOR: " + ex.getMessage());
    }
}
```

Método crearAutorConLibros

Resultado por Consola del método:

```
Hibernate:
    insert
    into
        autores
        (apellidos, fecha_nacimiento, nacionalidad, nombre)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        libros
        (id_autor, fecha_publicacion, isbn, numero_paginas, titulo)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        ejemplares
        (codigo_ejemplar, estado, id_libro, ubicacion)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        ejemplares
        (codigo_ejemplar, estado, id_libro, ubicacion)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        libros
        (id_autor, fecha_publicacion, isbn, numero_paginas, titulo)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        ejemplares
        (codigo_ejemplar, estado, id_libro, ubicacion)
    values
        (?, ?, ?, ?)
```

Resultado en la base de datos:

id_autor	nombre	apellidos	nacionalidad	fecha_nacimiento
1	Gabriel	García Márquez	Colombiana	1927-03-06
2	Isabel	Allende	Chilena	1942-08-02
3	Miguel	de Cervantes	Española	1547-09-29
4	Javier	Marías	Española	1951-09-20
NULL	NULL	NULL	NULL	NULL

4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
6	EJ-009-2024	DISPONIBLE	Estantería E-10	5

4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	3
5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
NULL	NULL	NULL	NULL	NULL	NULL

Mientras que la agregación de un nuevo ejemplar resultaría en lo siguiente:

```java
public void agregarEjemplarALibroExistente(String isbn, String codigoEjemplar) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el libro por el ISBN
        String hql = "FROM Libro WHERE isbn = :isbn";
        Query<Libro> query = session.createQuery(hql, Libro.class);
        query.setParameter("isbn", isbn); // Añade el isbn al parámetro
        Libro libro = query.uniqueResult(); // Obtiene el resultado de la consulta

        // Comprueba que el libro no sea null
        if (libro != null) {
            // Crea y añade un nuevo ejemplar
            Ejemplar nuevoEjemplar = new Ejemplar(codigoEjemplar, EstadoEjemplar.DISPONIBLE, "Almacén"); /
            libro.addEjemplar(nuevoEjemplar); // Lo añade al libro

            // Actualiza el libro existente
            session.merge(libro);
            // Hace los cambios permanentes y visibles
            tx.commit();
            System.out.println("El ejemplar se ha añadido al libro: " + libro.getTitulo());
        } else {
            // Si el libro es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado ningún libro con el ISBN: " + isbn);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL AGREGAR EL EJEMPLAR: " + ex.getMessage());
    }
}
```

Método agregarEjemplarALibroExistente

Resultado por Consola del método:

```
--- AGREGAR EJEMPLAR A LIBRO ---
Ingresa el ISBN del libro: 978-84-204-2955-0
Ingresa el c◆digo del ejemplar: EJ-012-2025
Hibernate:
    select
        libro0_.id_libro as id_libro1_2_,
        libro0_.id_autor as id_autor6_2_,
        libro0_.fecha_publicacion as fecha_pu2_2_,
        libro0_.isbn as isbn3_2_,
        libro0_.numero_paginas as numero_p4_2_,
        libro0_.titulo as titulo5_2_
    from
        libros libro0_
    where
        libro0_.isbn=?
Hibernate:
    insert
    into
        ejemplares
        (codigo_ejemplar, estado, id_libro, ubicacion)
    values
        (?, ?, ?, ?)
El ejemplar se ha a◆adido al libro: Ma◆ana en la batalla piensa en m◆
```

Resultado en la base de datos:

	8	EJ-011-2024	DISPONIBLE	Estantería E-11	6
	9	EJ-012-2025	DISPONIBLE	Almacén	6
*	NULL	NULL	NULL	NULL	NULL

Apartado 4: Operaciones Read

4.1 Explicación del Funcionamiento

Realicé diversas consultas empleando tanto HQL. El método listarTodosLosAutores recupera todas las instancias de Autor. A su vez, el método buscarLibroPorId usa session.get para una búsqueda rápida por clave primaria y carga los ejemplares sólo cuando se llama a libro.getEjemplares(). Por otro lado, buscarEjemplaresPorEstado filtra por un parámetro de HQL y obtenerEstadisticasBiblioteca usa funciones de agregación HQL (COUNT()) para obtener los totales.

4.2 Fragmentos de Código Relevantes con Resultado

A continuación, se detalla la estructura de cada método de consulta. Empezando por el listado de autores:

```java
public void listarTodosLosAutores() {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        // Lista todos los autores
        String hql = "FROM Autor";
        Query<Autor> query = session.createQuery(hql, Autor.class);
        List<Autor> autores = query.list(); // Obtiene el resultado de todos los autores

        // Itinera y muestra todos los autores por pantalla
        for (Autor autor : autores) {
            System.out.println("Autor[ID: " + autor.getIdAutor() + ", Nombre: " + autor.getNombre() + " " + autor.getApellidos()
                    + ", Nacionalidad: " + autor.getNacionalidad() + ", N° de Libros: " + autor.getLibros().size() + "]");
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        System.err.println("ERROR AL MOSTRAR LOS AUTORES: " + ex.getMessage());
    }
}
```

Método listarTodosLosAutores

Resultado por Consola del método:

```
--- LISTADO DE AUTORES ---
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_,
        autor0_.apellidos as apellido2_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_,
        autor0_.nacionalidad as nacional4_0_,
        autor0_.nombre as nombre5_0_
    from
        autores autor0_
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Autor[ID: 1, Nombre: Gabriel Garc♦a M♦rquez, Nacionalidad: Colombiana, N♦ de Libros: 2]
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Autor[ID: 2, Nombre: Isabel Allende, Nacionalidad: Chilena, N♦ de Libros: 1]
Hibernate:
```

Por otro lado, listar la información de un libro:

```
public void buscarLibroPorId(Integer idLibro) {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        // Obtiene el libro que concuerde con la id añadida
        Libro libro = session.get(Libro.class, idLibro);

        // Comprueba que el libro no sea null
        if (libro != null) {
            System.out.println("\n--- INFORMACIÓN DEL LIBRO ---");
            // Muestra la información del libro y autor
            System.out.println("Libro[Titulo: " + libro.getTitulo() + ", ISBN: " + libro.getIsbn() + ", Fecha de Publicación: " + libro.getFechaPublicacion()
                    + ", Nº de Páginas: " + libro.getNumeroPaginas() + "\n- Autor[ID: " + libro.getAutor().getIdAutor() + ", Nombre: " + libro.getAutor().getNombre()
                    + " " + libro.getAutor().getApellidos() + ", Nacionalidad: " + libro.getAutor().getNacionalidad() + ", Nº de Libros: "
                    + libro.getAutor().getLibros().size() + "]" + "\n- Número de Ejemplares: ");
            // Itinera y muestra todos los ejemplares
            for (Ejemplar e : libro.getEjemplares()) {
                System.out.println("  - Ejemplar[ID: " + e.getIdEjemplar() + ", Código: " + e.getCodigoEjemplar() + ", Estado: " + e.getEstado()
                        + ", Ubicación: " + e.getUbicacion() + "]");
            }
        } else {
            // Si el libro es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado ningún libro con el ID: " + idLibro);
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        System.err.println("ERROR AL BUSCAR EL LIBRO: " + ex.getMessage());
    }
}
```

Método buscarLibroPorId

Resultado por Consola del método:

```
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_0_,
        autor0_.apellidos as apellido2_0_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_0_,
        autor0_.nacionalidad as nacional4_0_0_,
        autor0_.nombre as nombre5_0_0_
    from
        autores autor0_
    where
        autor0_.id_autor=?
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Libro[T□tulo: Ma□ana en la batalla piensa en m□, ISBN: 978-84-204-2955-0, Fecha de Publicaci□n: 1994-05-05, N□ de P□ginas: 464
- Autor[ID: 4, Nombre: Javier Mar□as, Nacionalidad: Espa□ola, N□ de Libros: 2]
- N□mero de Ejemplares:
Hibernate:
    select
        ejemplares0_.id_libro as id_libro5_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_1_,
        ejemplares0_.codigo_ejemplar as codigo_e2_1_1_,
        ejemplares0_.estado as estado3_1_1_,
        ejemplares0_.id_libro as id_libro5_1_1_,
        ejemplares0_.ubicacion as ubicacio4_1_1_
    from
        ejemplares ejemplares0_
    where
        ejemplares0_.id_libro=?
  - Ejemplar[ID: 8, C□digo: EJ-011-2024, Estado: DISPONIBLE, Ubicaci□n: Estanter□a E-11]
  - Ejemplar[ID: 9, C□digo: EJ-012-2025, Estado: DISPONIBLE, Ubicaci□n: Almac□n]
```

La búsqueda por estado se vería de la siguiente forma:

```java
public void buscarEjemplaresPorEstado(EstadoEjemplar estado) {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        // Busca los ejemplares por estado
        String hql = "FROM Ejemplar e WHERE e.estado = :estado";
        Query<Ejemplar> query = session.createQuery(hql, Ejemplar.class);
        query.setParameter("estado", estado); // Añade el estado al parámetro
        List<Ejemplar> ejemplares = query.list(); // Obtiene el resultado de la consulta

        System.out.println("\n--- EJEMPLARES CON ESTADO: " + estado + " ---");
        // Itinera y muestra todos los ejemplares con el estado seleccionado por pantalla
        for (Ejemplar e : ejemplares) {
            System.out.println("Ejemplar[Código: " + e.getCodigoEjemplar() + ", Estado: " + e.getEstado() + ", Ubicación: "
                    + e.getUbicacion() + ", Título del Libro: " + e.getLibro().getTitulo() + "]");
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        System.err.println("ERROR AL BUSCAR LOS EJEMPLARES: " + ex.getMessage());
    }
}
```

Método buscarEjemplaresPorEstado

Resultado por Consola del método:

```
--- BUSCAR EJEMPLARES POR ESTADO ---

Selecciona un estado (1.Disponible, 2.Prestado, 3.Reparaci□n, 4.Baja): 2
Hibernate:
    select
        ejemplar0_.id_ejemplar as id_ejempl1_1_,
        ejemplar0_.codigo_ejemplar as codigo_e2_1_,
        ejemplar0_.estado as estado3_1_,
        ejemplar0_.id_libro as id_libro5_1_,
        ejemplar0_.ubicacion as ubicacio4_1_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.estado=?

--- EJEMPLARES CON ESTADO: PRESTADO ---
Hibernate:
    select
        libro0_.id_libro as id_libro1_2_0_,
        libro0_.id_autor as id_autor6_2_0_,
        libro0_.fecha_publicacion as fecha_pu2_2_0_,
        libro0_.isbn as isbn3_2_0_,
        libro0_.numero_paginas as numero_p4_2_0_,
        libro0_.titulo as titulo5_2_0_
    from
        libros libro0_
    where
        libro0_.id_libro=?
Ejemplar[C□digo: EJ-003-2024, Estado: PRESTADO, Ubicaci□n: Estanter□a A-13, T□tulo del Libro: El amor en los tiempos del c□lera]
```

Este apartado final presenta las estadísticas de la biblioteca. Dichas estadísticas desglosan los totales de autores, libros y ejemplares, además de clasificar los ejemplares según su estado.

```java
public void obtenerEstadisticasBiblioteca() {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        // Obtiene el total de autores únicos registrados en la BBDD
        Long totalAutores = session.createQuery("SELECT COUNT(a) FROM Autor a", Long.class).uniqueResult();

        // Obtiene el total de libros únicos registrados en la BBDD
        Long totalLibros = session.createQuery("SELECT COUNT(l) FROM Libro l", Long.class).uniqueResult();

        // Obtiene el total de ejemplares únicos registrados en la BBDD
        Long totalEjemplares = session.createQuery("SELECT COUNT(e) FROM Ejemplar e", Long.class).uniqueResult();

        // Muestra el total de cada consulta por pantalla
        System.out.println("Totales[Autores: " + totalAutores + " - Libros: " + totalLibros + " - Ejemplares: " + totalEjemplares + "]");
        // Muestra el total de ejemplares por estado
        System.out.println("\nEjemplares por estado:");
        for (EstadoEjemplar estado : EstadoEjemplar.values()) { // Itinera cada estado
            String hql = "SELECT COUNT(e) FROM Ejemplar e WHERE e.estado = :estado"; // Consulta
            // Añade el estado al parámetro e indica que deben ser valores únicos
            Long count = session.createQuery(hql, Long.class).setParameter("estado", estado).uniqueResult();
            System.out.println(" - " + estado + ": " + count); // Muestra el resultado
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        System.err.println("ERROR AL OBTENER LAS ESTADÍSTICAS: " + ex.getMessage());
    }
}
```

Método obtenerEstadisticasBiblioteca

Resultado por Consola del método:

```
--- ESTAD◆STICAS DE LA BIBLIOTECA ---
Hibernate:
    select
        count(autor0_.id_autor) as col_0_0_
    from
        autores autor0_
Hibernate:
    select
        count(libro0_.id_libro) as col_0_0_
    from
        libros libro0_
Hibernate:
    select
        count(ejemplar0_.id_ejemplar) as col_0_0_
    from
        ejemplares ejemplar0_
Totales[Autores: 4 - Libros: 6 - Ejemplares: 9]

Ejemplares por estado:
Hibernate:
    select
        count(ejemplar0_.id_ejemplar) as col_0_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.estado=?
 - DISPONIBLE: 8
Hibernate:
    select
        count(ejemplar0_.id_ejemplar) as col_0_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.estado=?
 - PRESTADO: 1
Hibernate:
    select
        count(ejemplar0_.id_ejemplar) as col_0_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.estado=?
 - REPARACION: 0
Hibernate:
    select
        count(ejemplar0_.id_ejemplar) as col_0_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.estado=?
 - BAJA: 0
```

Apartado 5: Operaciones Update

5.1 Explicación del Funcionamiento

Implementé funcionalidades para la modificación de datos ya existentes. Los métodos actualizarEstadoEjemplar y actualizarDatosLibro modifican sus atributos y tx.commit() se encarga de sincronizar los cambios con la BBDD (generando un UPDATE). El método transferirLibrosEntreAutores es más complejo, ya que debe cambiar los libros de autor.

5.2 Estructura de la Base de datos (Antes)

id_autor	nombre	apellidos	nacionalidad	fecha_nacimiento
1	Gabriel	García Márquez	Colombiana	1927-03-06
2	Isabel	Allende	Chilena	1942-08-02
3	Miguel	de Cervantes	Española	1547-09-29
4	Javier	Marías	Española	1951-09-20
NULL	NULL	NULL	NULL	NULL

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
2	El amor en los tiempos del cólera	978-84-397-0428-7	1985-09-05	464	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	3
5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
NULL	NULL	NULL	NULL	NULL	NULL

	id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
▶	1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
	2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
	3	EJ-003-2024	PRESTADO	Estantería A-13	2
	4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
	5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
	6	EJ-009-2024	DISPONIBLE	Estantería E-10	5
	7	EJ-010-2024	DISPONIBLE	Estantería E-10	5
	8	EJ-011-2024	DISPONIBLE	Estantería E-11	6
	9	EJ-012-2025	DISPONIBLE	Almacén	6
*	NULL	NULL	NULL	NULL	NULL

5.3 Fragmentos de Código Relevantes con Resultado

A continuación, se detalla la estructura de cada método de actualización. Empezando por el estado de un ejemplar:

```
public void actualizarEstadoEjemplar(Integer idEjemplar, EstadoEjemplar nuevoEstado) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el ejemplar por el ID
        Ejemplar ejemplar = session.get(Ejemplar.class, idEjemplar);

        // Comprueba que el ejemplar no sea null
        if (ejemplar != null) {
            // Añade a una variable el estado actual a la hora de la consulta
            EstadoEjemplar estadoAnterior = ejemplar.getEstado();
            ejemplar.setEstado(nuevoEstado); // Le asigna al ejemplar el nuevo estado

            // Si el nuevo estado es PRESTADO cambia la ubicación
            if (nuevoEstado == EstadoEjemplar.PRESTADO) {
                ejemplar.setUbicacion("En préstamo");
            }

            // Actualiza el ejemplar existente
            session.merge(ejemplar);
            // Hace los cambios permanentes y visibles
            tx.commit();

            // Muestra una confirmación del cambio de estado
            System.out.println("El estado  del ejemplar " + idEjemplar + " ha sido actualizado de " + estadoAnterior + " a " + nuevoEstado);
        } else {
            // Si el ejemplar es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado el ejemplar con el ID: " + idEjemplar);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL ACTUALIZAR EL ESTADO: " + ex.getMessage());
    }
}
```

Método actualizarEstadoEjemplar

Resultado por Consola del método:

```
--- ACTUALIZAR ESTADO DEL EJEMPLAR ---
Ingresa el ID del ejemplar: 8

Selecciona el nuevo estado del ejemplar (1.Disponible, 2.Prestado, 3.Reparaci□n, 4.Baja): 4
Hibernate:
    select
        ejemplar0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplar0_.codigo_ejemplar as codigo_e2_1_0_,
        ejemplar0_.estado as estado3_1_0_,
        ejemplar0_.id_libro as id_libro5_1_0_,
        ejemplar0_.ubicacion as ubicacio4_1_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.id_ejemplar=?
Hibernate:
    update
        ejemplares
    set
        codigo_ejemplar=?,
        estado=?,
        id_libro=?,
        ubicacion=?
    where
        id_ejemplar=?
El estado  del ejemplar 8 ha sido actualizado de DISPONIBLE a BAJA
```

Resultado en la base de datos:

id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
3	EJ-003-2024	PRESTADO	Estantería A-13	2
4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
6	EJ-009-2024	DISPONIBLE	Estantería E-10	5
7	EJ-010-2024	DISPONIBLE	Estantería E-10	5
8	EJ-011-2024	BAJA	Estantería E-11	6
9	EJ-012-2025	DISPONIBLE	Almacén	6
NULL	NULL	NULL	NULL	NULL

La actualización de los datos de los libros se presentaría de la siguiente forma:

```java
public void actualizarDatosLibro(Integer idLibro, String nuevoTitulo, Integer nuevasPaginas) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el libro por el ID
        Libro libro = session.get(Libro.class, idLibro);

        // Comprueba que el libro no sea null
        if (libro != null) {
            // Muestra un mensaje con los datos anteriores
            System.out.println("- Datos Anteriores[Título: " + libro.getTitulo() + " - Nº de Páginas: " + libro.getNumeroPaginas() + "]");

            // Asigna el nuevo título y número de páginas al libro
            libro.setTitulo(nuevoTitulo);
            libro.setNumeroPaginas(nuevasPaginas);

            // Actualiza el libro
            session.merge(libro);
            // Hace los cambios permanentes y visibles
            tx.commit();

            // Muestra un mensaje con los nuevos datos
            System.out.println("- Datos Actualizados[Título: " + nuevoTitulo + " - Nº de Páginas: " + nuevasPaginas + "]");
        } else {
            // Si el libro es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado el libro con el ID: " + idLibro);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL ACTUALIZAR EL LIBRO: " + ex.getMessage());
    }
}
```

Método actualizarDatosLibro

Resultado por Consola del método:

```
--- ACTUALIZAR DATOS DE LIBRO ---
Ingresa el ID del libro: 2
Ingresa el nuevo t□tulo: Pedritos House
Ingresa el nuevo n□mero de p□ginas: 18
Hibernate:
    select
        libro0_.id_libro as id_libro1_2_0_,
        libro0_.id_autor as id_autor6_2_0_,
        libro0_.fecha_publicacion as fecha_pu2_2_0_,
        libro0_.isbn as isbn3_2_0_,
        libro0_.numero_paginas as numero_p4_2_0_,
        libro0_.titulo as titulo5_2_0_
    from
        libros libro0_
    where
        libro0_.id_libro=?
- Datos Anteriores[T□tulo: PEDRITOS HOUSE - N□ de P□ginas: 666]
Hibernate:
    update
        libros
    set
        id_autor=?,
        fecha_publicacion=?,
        isbn=?,
        numero_paginas=?,
        titulo=?
    where
        id_libro=?
- Datos Actualizados[T□tulo: Pedritos House - N□ de P□ginas: 18]
```

Resultado en la base de datos:

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
2	Pedritos House	978-84-397-0428-7	1985-09-05	18	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	3
5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
NULL	NULL	NULL	NULL	NULL	NULL

Finalmente, he implementado la transferencia de libros entre autores, asegurando la consistencia bidireccional, de la siguiente manera:

```java
public void transferirLibrosEntreAutores(Integer idAutorOrigen, Integer idAutorDestino) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el autor de Origen y de Destino por el ID
        Autor autorOrigen = session.get(Autor.class, idAutorOrigen);
        Autor autorDestino = session.get(Autor.class, idAutorDestino);

        // Comprueba que los autores no sean null
        if (autorOrigen != null && autorDestino != null) {
            // Copia los libros a transferir
            List<Libro> librosATransferir = new ArrayList<>(autorOrigen.getLibros());
            int cantidadLibros = librosATransferir.size(); // Obtiene la cantidad de libros

            // Itinera cada libro, cambiando el autor de libro
            for (Libro libro : librosATransferir) {
                libro.setAutor(autorDestino);
                session.merge(libro); // Actualiza el libro
            }

            // Hace los cambios permanentes y visibles
            tx.commit();

            // Muestra la cantidad de libros transferidos de un autor a otro
            System.out.println("Se han transferido " + cantidadLibros + " libros del autor " + autorOrigen.getNombre()
                    + " " + autorOrigen.getApellidos() + " al autor " + autorDestino.getNombre() + " " + autorDestino.getApellidos());
        } else {
            // Si el autor es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado a uno o ambos autores.");
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL TRANSFERIR LOS LIBROS: " + ex.getMessage());
    }
}
```

Método transferirLibrosEntreAutores

Resultado por Consola del método:

```
--- TRANSFERIR LIBROS ENTRE AUTORES ---
Ingresa el ID del autor de ORIGEN: 3
Ingresa el ID del autor de DESTINO: 4
```

```
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_0_,
        autor0_.apellidos as apellido2_0_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_0_,
        autor0_.nacionalidad as nacional4_0_0_,
        autor0_.nombre as nombre5_0_0_
    from
        autores autor0_
    where
        autor0_.id_autor=?
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_0_,
        autor0_.apellidos as apellido2_0_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_0_,
        autor0_.nacionalidad as nacional4_0_0_,
        autor0_.nombre as nombre5_0_0_
    from
        autores autor0_
    where
        autor0_.id_autor=?
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Hibernate:
    update
        libros
    set
        id_autor=?,
        fecha_publicacion=?,
        isbn=?,
        numero_paginas=?,
        titulo=?
    where
        id_libro=?
Se han transferido 1 libros del autor Miguel de Cervantes al autor Javier Mar❖as
```

Resultado en la base de datos:

	id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
▶	1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
	2	Pedritos House	978-84-397-0428-7	1985-09-05	18	1
	3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
	4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	4
	5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
	6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
*	NULL	NULL	NULL	NULL	NULL	NULL

Apartado 6: Operaciones Delete

6.1 Explicación del Funcionamiento

Implementé métodos de borrado, prestando especial atención a la consistencia bidireccional y al funcionamiento del borrado en cascada. Para eliminarEjemplar y eliminarLibro, es crucial llamar primero al método helper (libro.removeEjemplar(...) o autor.removeLibro(...)) para desvincular el objeto de la entidad padre en el lado de Java. Después, session.remove() lo elimina. En eliminarAutorConCascada, simplemente se llama a session.remove(autor), y la configuración CascadeType.ALL y orphanRemoval=true se encarga de eliminar todos los libros y ejemplares asociados a ese autor automáticamente.

6.2 Estructura de la Base de datos (Antes)

id_autor	nombre	apellidos	nacionalidad	fecha_nacimiento
1	Gabriel	García Márquez	Colombiana	1927-03-06
2	Isabel	Allende	Chilena	1942-08-02
3	Miguel	de Cervantes	Española	1547-09-29
4	Javier	Marías	Española	1951-09-20
NULL	NULL	NULL	NULL	NULL

id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
3	EJ-003-2024	PRESTADO	Estantería A-13	2
4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
6	EJ-009-2024	DISPONIBLE	Estantería E-10	5
7	EJ-010-2024	DISPONIBLE	Estantería E-10	5
8	EJ-011-2024	BAJA	Estantería E-11	6
9	EJ-012-2025	DISPONIBLE	Almacén	6
NULL	NULL	NULL	NULL	NULL

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
2	Pedritos House	978-84-397-0428-7	1985-09-05	18	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	4
5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
NULL	NULL	NULL	NULL	NULL	NULL

6.3 Fragmentos de Código Relevantes con Resultado

A continuación, se detalla la estructura de cada método de eliminación. Empezando por la eliminación de un ejemplar por ID:

```java
public void eliminarEjemplar(Integer idEjemplar) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el ejemplar por el ID
        Ejemplar ejemplar = session.get(Ejemplar.class, idEjemplar);

        // Comprueba que el ejemplar no sea null
        if (ejemplar != null) {
            // Obtiene el código de ejemplar y la información del libro
            String codigo = ejemplar.getCodigoEjemplar();
            Libro libro = ejemplar.getLibro();

            // Mantiene la consistencia bidireccional, utilizando el método de la clase libro
            libro.removeEjemplar(ejemplar);

            // Elimina el ejemplar existente
            session.remove(ejemplar);
            // Hace los cambios permanentes y visibles
            tx.commit();

            // Muestra un mensaje de confirmación del borrado
            System.out.println("El ejemplar ha sido eliminado: " + codigo);
        } else {
            // Si el ejemplar es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado el ejemplar con el ID: " + idEjemplar);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL ELIMINAR EL EJEMPLAR: " + ex.getMessage());
    }
}
```

Método eliminarEjemplar

Resultado por Consola del método:

```
--- ELIMINAR EJEMPLAR ---
Ingresa el ID del ejemplar a eliminar: 8
�Est�s seguro? (S/N): s
Hibernate:
    select
        ejemplar0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplar0_.codigo_ejemplar as codigo_e2_1_0_,
        ejemplar0_.estado as estado3_1_0_,
        ejemplar0_.id_libro as id_libro5_1_0_,
        ejemplar0_.ubicacion as ubicacio4_1_0_
    from
        ejemplares ejemplar0_
    where
        ejemplar0_.id_ejemplar=?
Hibernate:
    select
        libro0_.id_libro as id_libro1_2_0_,
        libro0_.id_autor as id_autor6_2_0_,
        libro0_.fecha_publicacion as fecha_pu2_2_0_,
        libro0_.isbn as isbn3_2_0_,
        libro0_.numero_paginas as numero_p4_2_0_,
        libro0_.titulo as titulo5_2_0_
    from
        libros libro0_
    where
        libro0_.id_libro=?
Hibernate:
    select
        ejemplares0_.id_libro as id_libro5_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_1_,
        ejemplares0_.codigo_ejemplar as codigo_e2_1_1_,
        ejemplares0_.estado as estado3_1_1_,
        ejemplares0_.id_libro as id_libro5_1_1_,
        ejemplares0_.ubicacion as ubicacio4_1_1_
    from
        ejemplares ejemplares0_
    where
        ejemplares0_.id_libro=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
El ejemplar ha sido eliminado: EJ-011-2024
```

Resultado en la base de datos:

id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
3	EJ-003-2024	PRESTADO	Estantería A-13	2
4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
5	EJ-005-2024	DISPONIBLE	Estantería C-20	4
6	EJ-009-2024	DISPONIBLE	Estantería E-10	5
7	EJ-010-2024	DISPONIBLE	Estantería E-10	5
9	EJ-012-2025	DISPONIBLE	Almacén	6
NULL	NULL	NULL	NULL	NULL

A continuación, se muestra la implementación de la eliminación de un libro:

```java
public void eliminarLibro(Integer idLibro) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el libro por el ID
        Libro libro = session.get(Libro.class, idLibro);

        // Comprueba que el libro no sea null
        if (libro != null) {
            // Obtiene el título del libro, número de ejemplares y el autor
            String titulo = libro.getTitulo();
            int numEjemplares = libro.getEjemplares().size();
            Autor autor = libro.getAutor();

            // Mantiene la consistencia bidireccional, utilizando el método de la clase autor
            autor.removeLibro(libro);

            // Elimina el libro existente
            session.remove(libro);
            // Hace los cambios permanentes y visibles
            tx.commit();

            System.out.println("El libro con el título '" + titulo + "' ha sido eliminado, junto a los " + numEjemplares + " ejemplares.");
        } else {
            // Si el libro es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado el libro con el ID: " + idLibro);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("Error al eliminar libro: " + ex.getMessage());
    }
}
```

Método eliminarLibro

Resultado por Consola del método:

```
--- ELIMINAR LIBRO ---
Ingresa el ID del libro a eliminar: 2
�Est� seguro? (S/N): s
Hibernate:
    select
        libro0_.id_libro as id_libro1_2_0_,
        libro0_.id_autor as id_autor6_2_0_,
        libro0_.fecha_publicacion as fecha_pu2_2_0_,
        libro0_.isbn as isbn3_2_0_,
        libro0_.numero_paginas as numero_p4_2_0_,
        libro0_.titulo as titulo5_2_0_
    from
        libros libro0_
    where
        libro0_.id_libro=?
Hibernate:
    select
        ejemplares0_.id_libro as id_libro5_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl_1_1_,
        ejemplares0_.codigo_ejemplar as codigo_e2_1_1_,
        ejemplares0_.estado as estado3_1_1_,
        ejemplares0_.id_libro as id_libro5_1_1_,
        ejemplares0_.ubicacion as ubicacio4_1_1_
    from
        ejemplares ejemplares0_
    where
        ejemplares0_.id_libro=?
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_0_,
        autor0_.apellidos as apellido2_0_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_0_,
        autor0_.nacionalidad as nacional4_0_0_,
        autor0_.nombre as nombre5_0_0_
    from
        autores autor0_
    where
        autor0_.id_autor=?
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
Hibernate:
    delete
    from
        libros
    where
        id_libro=?
El libro con el t�tulo 'Pedritos House' ha sido eliminado, junto a los 1 ejemplares.
```

Resultado en la base de datos:

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
4	Don Quijote de la Mancha	978-84-667-0814-4	1605-01-16	863	4
5	Corazón tan blanco	978-84-204-2954-3	1992-03-01	368	4
6	Mañana en la batalla piensa en mí	978-84-204-2955-0	1994-05-05	464	4
NULL	NULL	NULL	NULL	NULL	NULL

Finalmente, la eliminación de un autor se representaría de la siguiente manera:

```
public void eliminarAutorConCascada(Integer idAutor) {
    Transaction tx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        tx = session.beginTransaction();

        // Busca el autor por el ID
        Autor autor = session.get(Autor.class, idAutor);

        // Comprueba que el autor no sea null
        if (autor != null) {
            // Obtiene el nombre y apellidos junto al número de libros del autor
            String nombreCompleto = autor.getNombre() + " " + autor.getApellidos();
            int numLibros = autor.getLibros().size();

            // Cuenta el número de ejemplares
            int numEjemplares = 0;
            for (Libro libro : autor.getLibros()) {
                numEjemplares += libro.getEjemplares().size();
            }

            // Elimina el autor existente
            session.remove(autor);
            // Hace los cambios permanentes y visibles
            tx.commit();

            // Muestra un mensaje de confirmación del borrado
            System.out.println("Se ha eliminado el autor " + nombreCompleto + " con "
                    + numLibros + " libros y " + numEjemplares + " ejemplares.");
        } else {
            // Si el autor es null, muestra un mensaje de error y hace rollback
            System.out.println("No se ha encontrado el autor con ID: " + idAutor);
            tx.rollback();
        }
        // Captura las posibles excepciones
    } catch (Exception ex) {
        // En caso de que la transacción sea diferente a null, realiza un rollback
        if (tx != null) {
            tx.rollback();
        }
        System.err.println("ERROR AL ELIMINAR EL AUTOR: " + ex.getMessage());
    }
}
```

Método eliminarAutorConCascada

Resultado por Consola del método:

```
--- ELIMINAR AUTOR ---
Ingresa el ID del autor a eliminar: 4
�Est� seguro? (S/N): s
Hibernate:
    select
        autor0_.id_autor as id_autor1_0_0_,
        autor0_.apellidos as apellido2_0_0_,
        autor0_.fecha_nacimiento as fecha_na3_0_0_,
        autor0_.nacionalidad as nacional4_0_0_,
        autor0_.nombre as nombre5_0_0_
    from
        autores autor0_
    where
        autor0_.id_autor=?
Hibernate:
    select
        libros0_.id_autor as id_autor6_2_0_,
        libros0_.id_libro as id_libro1_2_0_,
        libros0_.id_libro as id_libro1_2_1_,
        libros0_.id_autor as id_autor6_2_1_,
        libros0_.fecha_publicacion as fecha_pu2_2_1_,
        libros0_.isbn as isbn3_2_1_,
        libros0_.numero_paginas as numero_p4_2_1_,
        libros0_.titulo as titulo5_2_1_
    from
        libros libros0_
    where
        libros0_.id_autor=?
Hibernate:
    select
        ejemplares0_.id_libro as id_libro5_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_1_,
        ejemplares0_.codigo_ejemplar as codigo_e2_1_1_,
        ejemplares0_.estado as estado3_1_1_,
        ejemplares0_.id_libro as id_libro5_1_1_,
        ejemplares0_.ubicacion as ubicacio4_1_1_
    from
        ejemplares ejemplares0_
    where
        ejemplares0_.id_libro=?
...
Hibernate:
    select
        ejemplares0_.id_libro as id_libro5_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_0_,
        ejemplares0_.id_ejemplar as id_ejempl1_1_1_,
        ejemplares0_.codigo_ejemplar as codigo_e2_1_1_,
        ejemplares0_.estado as estado3_1_1_,
        ejemplares0_.id_libro as id_libro5_1_1_,
        ejemplares0_.ubicacion as ubicacio4_1_1_
    from
        ejemplares ejemplares0_
    where
        ejemplares0_.id_libro=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
Hibernate:
    delete
    from
        libros
    where
        id_libro=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
Hibernate:
    delete
    from
        ejemplares
    where
        id_ejemplar=?
Hibernate:
    delete
    from
        libros
    where
        id_libro=?
Hibernate:
    delete
    from
        libros
    where
        id_libro=?
Hibernate:
    delete
    from
        autores
    where
        id_autor=?
Se ha eliminado el autor Javier Mar�as con 3 libros y 4 ejemplares.
```

Resultado en la base de datos:

id_autor	nombre	apellidos	nacionalidad	fecha_nacimiento
1	Gabriel	García Márquez	Colombiana	1927-03-06
2	Isabel	Allende	Chilena	1942-08-02
3	Miguel	de Cervantes	Española	1547-09-29
NULL	NULL	NULL	NULL	NULL

id_ejemplar	codigo_ejemplar	estado	ubicacion	id_libro
1	EJ-001-2024	DISPONIBLE	Estantería A-12	1
2	EJ-002-2024	DISPONIBLE	Estantería A-12	1
4	EJ-004-2024	DISPONIBLE	Estantería B-05	3
NULL	NULL	NULL	NULL	NULL

id_libro	titulo	isbn	fecha_publicacion	numero_paginas	id_autor
1	Cien años de soledad	978-84-376-0494-7	1967-06-05	471	1
3	La casa de los espíritus	978-84-204-2943-7	1982-10-01	433	2
NULL	NULL	NULL	NULL	NULL	NULL

Apartado 7: Clase principal y menú

7.1 Explicación del Funcionamiento

He creado una clase Main con un menú interactivo (con while y switch) para acceder a las funcionalidades de GestionBiblioteca. Implementé un bloque try-catch para manejar InputMismatchException y así validar la entrada numérica del usuario. Me aseguré de que la opción de salida (0) cierre correctamente todos los recursos, incluyendo HibernateUtil.shutdown() y el Scanner.

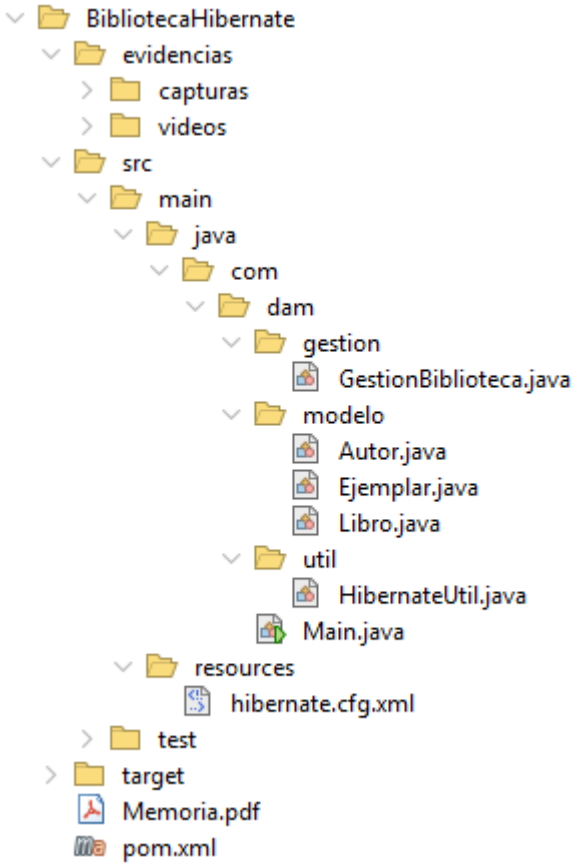
7.2 Fragmentos de Código Relevantes

```
public static void main(String[] args) {
    boolean salir = false; // Variable para controlar el bucle while

    // Bucle While: Muestra el menú y espera la opción del usuario.
    while (!salir) {
        try {
            System.out.print("\n--- SISTEMA DE GESTIÓN DE BIBLIOTECA ---\n1.Crear nuevo autor con lib
            int opcion = sc.nextInt(); // Lee la opción del usuario

            // Switch: Ejecuta la acción correspondiente a la opción seleccionada
            switch (opcion) {
                case 1 ->
                    crearAutorConLibros(); // Crea un autor con sus libros
                case 2 ->
                    agregarEjemplarALibro(); // Agrega un ejemplar a un libro existente
                case 3 ->
                    listarAutores(); // Lista todos los autores
                case 4 ->
                    buscarLibro(); // Busca un libro por su ID
                case 5 ->
                    buscarEjemplaresPorEstado(); // Busca los ejemplares por su estado
                case 6 ->
                    mostrarEstadisticas(); // Muestra las estadísticas de la biblioteca
                case 7 ->
                    actualizarEstadoEjemplar(); // Actualiza el estado de un ejemplar
                case 8 ->
                    actualizarDatosLibro(); // Actualiza los datos de un libro
                case 9 ->
                    transferirLibrosEntreAutores(); // Transferir libros entre autores
                case 10 ->
                    eliminarEjemplar(); // Elimina un ejemplar
                case 11 ->
                    eliminarLibro(); // Elimina un libro
                case 12 ->
                    eliminarAutor(); // Elimina un autor con cascada
                case 0 ->
                    salir = true; // Cierra el bucle
                default ->
                    System.out.println("La opción seleccionada no es válida. Intentalo de nuevo.");
            }
            // Captura las posibles excepciones
        } catch (InputMismatchException e) {
            System.out.println("ERROR: Debe introducir un número válido.");
        } catch (Exception ex) {
            System.out.println("ERROR: " + ex.getMessage());
        }
    }

    // Cierra los recursos
    HibernateUtil.shutdown();
    sc.close();
    System.out.println("\nSe ha cerrado la sesión.");
}
```

Información Adicional

1. Estructura del Proyecto

El proyecto presenta la siguiente organización:



2. Dificultades encontradas y soluciones aplicadas

Las principales dificultades y soluciones implementadas en mi proyecto de gestión de biblioteca con Hibernate fueron:

2.1. Implementación de la transferencia de libros:

- Dificultad: Al intentar transferir un libro de un autor a otro, se producía un error de ejecución debido a la imposibilidad de realizar la transferencia utilizando los helpers que diseñé para tal fin.
- Solución: Para solucionar este inconveniente, opté por llamar directamente al método getter getAutor y realizar la modificación del autor del libro desde ese punto.

2.2. Eliminar Entidades con Relaciones Bidireccionales:

- Dificultad: Surgían errores de estado de Hibernate al intentar eliminar directamente un objeto libro o ejemplar usando session.remove(). Esto ocurría porque el objeto autor o libro aún mantenía una referencia al objeto en su colección.

- Solución: Para resolver esto, fue necesario desvincular el objeto hijo de su padre en el lado de Java antes de llamar a session.remove() sobre el objeto hijo. Para ello, utilicé los métodos autor.removeLibro(libro) y libro.removeEjemplar(ejemplar).

3. Conclusión

El proyecto culmina con la implementación de un sistema CRUD completo para la gestión de una biblioteca, utilizando Java y la potencia de Hibernate. La adaptación de Hibernate resultó crucial, simplificando drásticamente el acceso a los datos al eliminar la necesidad de escribir código SQL manual. La complejidad principal del desarrollo fue la configuración de las entidades y la gestión correcta del ciclo de vida de los objetos con las relaciones bidireccionales.