



AE-3 JPA

21/02/2022

Irene Alonso, Frida Abella, David Matías y Jaime Aranda

La idea de esta aplicación será la de generar un modelo de datos para gestionar una **cadena de librerías**.

GITHUB:


GitHub - JaimeArandaCongil/AE3-JPA

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/JaimeArandaCongil/AE3-JPA>

JaimeArandaCongil

/AE3-JPA



1

Contributor

0

Issues

0

Stars

0

Forks

Requerimiento 1

Se pide diseñar el programa mediante JPA que cumpla con, al menos, los siguientes requisitos y entidades:

1. Autor, tendrá un id, un nombre, unos apellidos y una fecha de nacimiento. Un autor podrá escribir muchos libros.

```
@Entity
public class Autor {

    // Asignamos al atributo id la cualidad de PrimaryKey o identificador único, y hacemos que se autogenera (autoincrementado)
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String nombre;
    private String apellidos;
    private String fechaNacimiento;

    @OneToMany(mappedBy="autor", cascade=CascadeType.PERSIST) // Le ponemos cascade persist para que dé de alta libros cuando demos de alta l
    // Relación bidireccional con la entidad Libro
    private List<Libro> librosEscritos;

    public Autor() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Autor(Integer id, String nombre, String apellidos, String fechaNacimiento) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.fechaNacimiento = fechaNacimiento;
    }

    public int getId() {
        return id;
    }
}
```

```

public void setId(Integer id) {
    this.id = id;
}
public String getNombre() {
    return nombre;
}
public void setNombre(String nombre) {
    this.nombre = nombre;
}
public String getApellidos() {
    return apellidos;
}
public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}
public String getFechaNacimiento() {
    return fechaNacimiento;
}
public void setFechaNacimiento(String fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}

public List<Libro> getLibrosEscritos() {
    return librosEscritos;
}

public void setLibrosEscritos(List<Libro> librosEscritos) {
    this.librosEscritos = librosEscritos;
}
@Override
public String toString() {
    return "Autor [id=" + id + ", nombre=" + nombre + ", apellidos=" + apellidos + ", fechaNacimiento="
        + fechaNacimiento + "]";
}
}

```

2. Editorial, tendrá un id, un nombre y una dirección. También tendrá una colección de libros publicados por la editorial.

```

@Entity
public class Editorial {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String nombre;
    private String direccion;

    // Referenciamos en los dos lados para que sea bidireccional
    @OneToMany(mappedBy="editorial", cascade=CascadeType.PERSIST) // Le ponemos cascade persist para que dé de alta libros cuando demos de al
    private List<Libro> librosPublicados;

    public Editorial() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Editorial(Integer id, String nombre, String direccion) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.direccion = direccion;
    }

    public int getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}

```

```

    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public List<Libro> getLibrosPublicados() {
        return librosPublicados;
    }

    public void setLibrosPublicados(List<Libro> librosPublicados) {
        this.librosPublicados = librosPublicados;
    }
}

```

3. Libro, tendrá un id, un título, un precio, una editorial y un autor.

```

@Entity
public class Libro {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String titulo;
    private double precio;

    @ManyToOne // No le pondremos un (cascade=CascadeType.ALL) para que borrar un libr no nos borre la editorial
    // Creamos la foreign key que unirá las dos tablas o entidades, que será el id del autor
    // Dado que estamos en el lado de "many" de la relación, la @JoinColumn siempre estará en este lado en este tipo de relaciones
    @JoinColumn(name="fk_id_autor", referencedColumnName="id")
    private Autor autor;

    @ManyToOne // No le pondremos un (cascade=CascadeType.ALL) para que borrar un libr no nos borre la editorial
    // Creamos la foreign key que unirá las dos tablas o entidades, que será el id de la editorial
    // Dado que estamos en el lado de "many" de la relación, la @JoinColumn siempre estará en este lado en este tipo de relaciones
    @JoinColumn(name="fk_id_editorial", referencedColumnName="id")
    private Editorial editorial;

    @ManyToMany(mappedBy="coleccionLibros", cascade=CascadeType.PERSIST)
    private List<Libreria> librerias;

    public Libro() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Libro(Integer id, String titulo, double precio, Autor autor, Editorial editorial, List<Libreria> librerias) {
        super();
        this.id = id;
        this.titulo = titulo;
        this.precio = precio;
        this.autor = autor;
        this.editorial = editorial;
        this.librerias = librerias;
    }

    public int getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public double getPrecio() {

```

```

        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public Autor getAutor() {
        return autor;
    }

    public void setAutor(Autor autor) {
        this.autor = autor;
    }

    public Editorial getEditorial() {
        return editorial;
    }

    public void setEditorial(Editorial editorial) {
        this.editorial = editorial;
    }

    public List<Libreria> getLibrerias() {
        return librerias;
    }

    public void setLibrerias(List<Libreria> librerias) {
        this.librerias = librerias;
    }
}

```

4. Librería, tendrá un id, un nombre, un nombre del dueño, una dirección y una colección de libros. Además, hay que tener en cuenta que un libro puede estar en diferentes librerías.

```

@Entity
public class Libreria {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String nombre;
    private String nombreDueño;
    private String direccion;

    // Establecemos la relación ManyToMany con la entidad libro, creando una tabla intermedia (librerias_libros) en la que
    // se conectan las entidades Libro y Libreria por medio de foreign keys (que corresponden con la P.K. o id de cada una de las entidades)
    @ManyToMany(cascade=CascadeType.PERSIST)
    @JoinTable(name="librerias_libros",
        joinColumns= {@JoinColumn(name="fk_id_libreria", referencedColumnName="id")},
        inverseJoinColumns= {@JoinColumn(name="fk_id_libro", referencedColumnName="id")})
    private List<Libro> coleccionLibros;

    public Libreria() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Libreria(Integer id, String nombre, String nombreDueño, String direccion, List<Libro> coleccionLibros) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.nombreDueño = nombreDueño;
        this.direccion = direccion;
        this.coleccionLibros = coleccionLibros;
    }

    public int getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }
}

```

```

    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getNombreDueño() {
        return nombreDueño;
    }

    public void setNombreDueño(String nombreDueño) {
        this.nombreDueño = nombreDueño;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public List<Libro> getColeccionLibros() {
        return coleccionLibros;
    }

    public void setColeccionLibros(List<Libro> coleccionLibros) {
        this.coleccionLibros = coleccionLibros;
    }
}

```

Una vez diseñado el modelo de datos y creadas las tablas se pide hacer las siguientes operaciones.

1. Dar de alta 3 autores.
2. Dar de alta 2 editoriales.
3. Dar de alta 8 libros, cada libro será escrito por uno de los autores dados de alta previamente y pertenecerá a uno de los editoriales dados de alta previamente.
4. 2 librerías, cada librería tendrá 4 libros dados de alta previamente.

```

// Creamos tres objetos de tipo Autor
Autor autor1 = new Autor(null, "Javier", "Marías", "20-09-1951");
Autor autor2 = new Autor(null, "Julio", "Verne", "08-02-1828");
Autor autor3 = new Autor(null, "Agatha", "Christie", "15-09-1890");

// Creamos dos objetos de tipo editorial
Editorial edi1 = new Editorial(null, "Espasa", "Calle Josefa Valcárcel, 42. Madrid");
Editorial edi2 = new Editorial(null, "Galaxia", "Avenida Madrid, 44. Vigo");

// Creamos los libros y les asignamos su autor y editorial
Libro libro1 = new Libro(null, "Los dominios del lobo", 25.5, autor1, edi2, null);
Libro libro2 = new Libro(null, "El monarca del tiempo", 15.9, autor1, edi1, null);
Libro libro3 = new Libro(null, "Viaje al centro de la tierra", 12.7, autor2, edi2, null);
Libro libro4 = new Libro(null, "La isla misteriosa", 30, autor2, edi2, null);
Libro libro5 = new Libro(null, "De la tierra a la luna", 28.3, autor2, edi1, null);
Libro libro6 = new Libro(null, "Muerte sobre el Nilo", 21, autor3, edi2, null);
Libro libro7 = new Libro(null, "Asesinato en el Orient Expres", 17.5, autor3, edi1, null);
Libro libro8 = new Libro(null, "Misterio en el caribe", 12, autor3, edi1, null);

//Añadimos los libros asociados al autor a una lista, no nos olvidamos de cruzar las
//referencias para hacerlo bidireccional, dado que a los libros ya les asignamos su autor al crearlos (relación Many to One)
List<Libro> librosJavierMarías = new ArrayList<Libro>();
librosJavierMarías.add(libro1);
librosJavierMarías.add(libro2);
//hacemos bidireccionalidad del autor Javier Marías y le asignamos sus lista de libros escritos
autor1.setLibrosEscritos(librosJavierMarías);

// Creamos la lista de libros de Julio Verne y se le asignamos al autor para que sea bidireccional la relación
List<Libro> librosJulioVerne = new ArrayList<Libro>();
librosJulioVerne.add(libro3);
librosJulioVerne.add(libro4);
librosJulioVerne.add(libro5);
autor2.setLibrosEscritos(librosJulioVerne);

```

```

// Creamos la lista de libros de Julio Verne y se los asignamos al autor (bidireccional)
List<Libro> librosAgathaChristie = new ArrayList<Libro>();
librosAgathaChristie.add(libro6);
librosAgathaChristie.add(libro7);
librosAgathaChristie.add(libro8);
autor3.setLibrosEscritos(librosAgathaChristie);

// Aquí asociaremos a cada editorial una lista o colección de libros (relación Many to One)
// Creamos una lista de libros que hayan sido publicados por cada editorial y se la asignamos a esa editorial (bidireccionalidad)
List<Libro> librosEditorialEspasa = new ArrayList<Libro>();
librosEditorialEspasa.add(libro2);
librosEditorialEspasa.add(libro5);
librosEditorialEspasa.add(libro7);
librosEditorialEspasa.add(libro8);
edi1.setLibrosPublicados(librosEditorialEspasa);
// Hacemos lo mismo con la segunda editorial:
List<Libro> librosEditorialGalaxia = new ArrayList<Libro>();
librosEditorialGalaxia.add(libro2);
librosEditorialGalaxia.add(libro5);
librosEditorialGalaxia.add(libro7);
librosEditorialGalaxia.add(libro8);
edi2.setLibrosPublicados(librosEditorialGalaxia);

// Creamos los objetos librería
Libreria libreria1 = new Libreria(null, "Casa del Libro", "Juan Pedro", "Calle Gran Vía, 29. Madrid", null);
Libreria libreria2 = new Libreria(null, "Librería Bardón", "Luisa Abeledo", "Plaza de San Martín, 3. Madrid", null);

// A continuación procederemos a crear las conexiones ManyToMany entre libros y librerías
// Para ello tendremos que crear una lista para cada librería y asignarle unos libros
// y una lista para cada libro y asignarle unas librerías (de manera que sea bidireccional)

// Comenzamos creando una lista de libros para la librería Casa del Libro, que tendrá 4 libros
List<Libro> librosCasaLibro = new ArrayList<Libro>();
librosCasaLibro.add(libro1);
librosCasaLibro.add(libro3);
librosCasaLibro.add(libro4);
librosCasaLibro.add(libro7);
// A continuación asignamos esa lista a la librería:
libreria1.setColeccionLibros(librosCasaLibro);

// Repetimos la operación con la otra editorial Libros Bardón, que tendrá otros 4 libros
List<Libro> librosBardon = new ArrayList<Libro>();
librosBardon.add(libro2);
librosBardon.add(libro3);
librosBardon.add(libro4);
librosBardon.add(libro7);
// A continuación asignamos esa lista a la librería:
libreria2.setColeccionLibros(librosBardon);

// BIDIRECCIONALIDAD:
// Haremos lo mismo a la inversa, asignando al objeto libro una lista de librerías (BIDIRECCIONALIDAD)

List<Libreria> libreriasLibro1 = new ArrayList<Libreria>();
libreriasLibro1.add(libreria1);
libro1.setLibrerias(libreriasLibro1);

List<Libreria> libreriasLibro2 = new ArrayList<Libreria>();
libreriasLibro2.add(libreria2);
libro2.setLibrerias(libreriasLibro2);

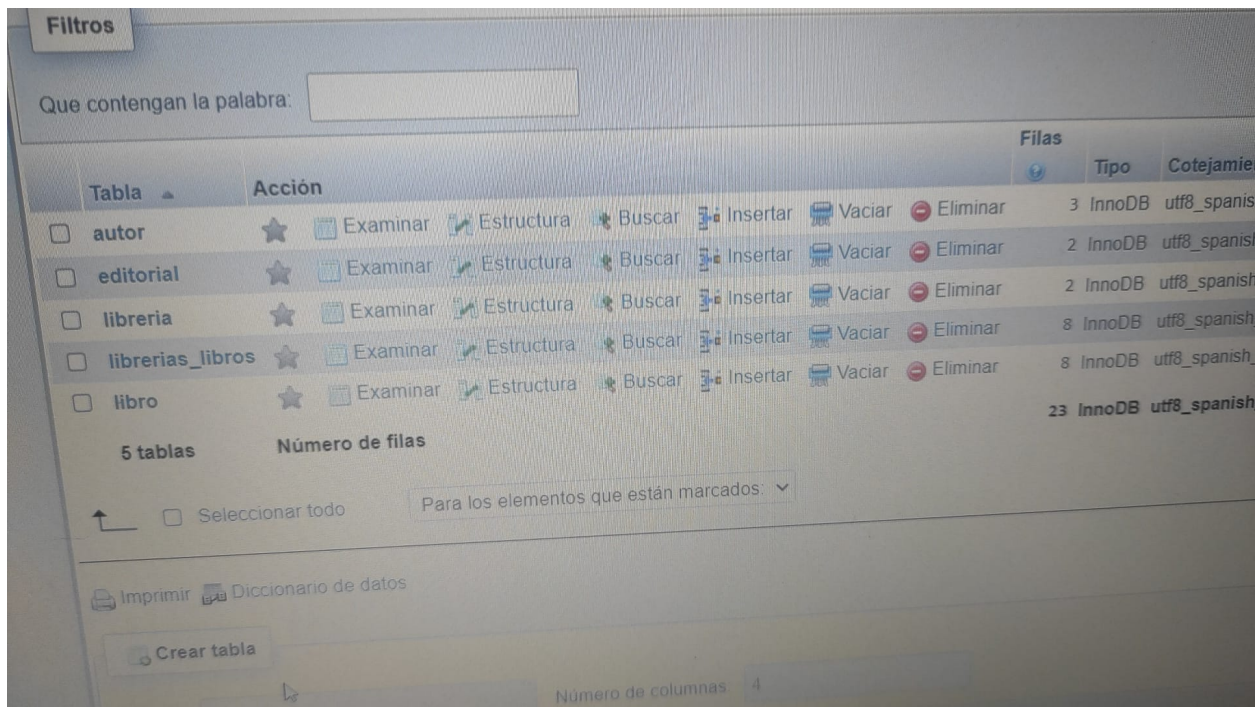
List<Libreria> libreriasLibro3 = new ArrayList<Libreria>();
libreriasLibro3.add(libreria1);
libreriasLibro3.add(libreria2);
libro3.setLibrerias(libreriasLibro3);

List<Libreria> libreriasLibro4 = new ArrayList<Libreria>();
libreriasLibro4.add(libreria1);
libreriasLibro4.add(libreria2);
libro4.setLibrerias(libreriasLibro4);

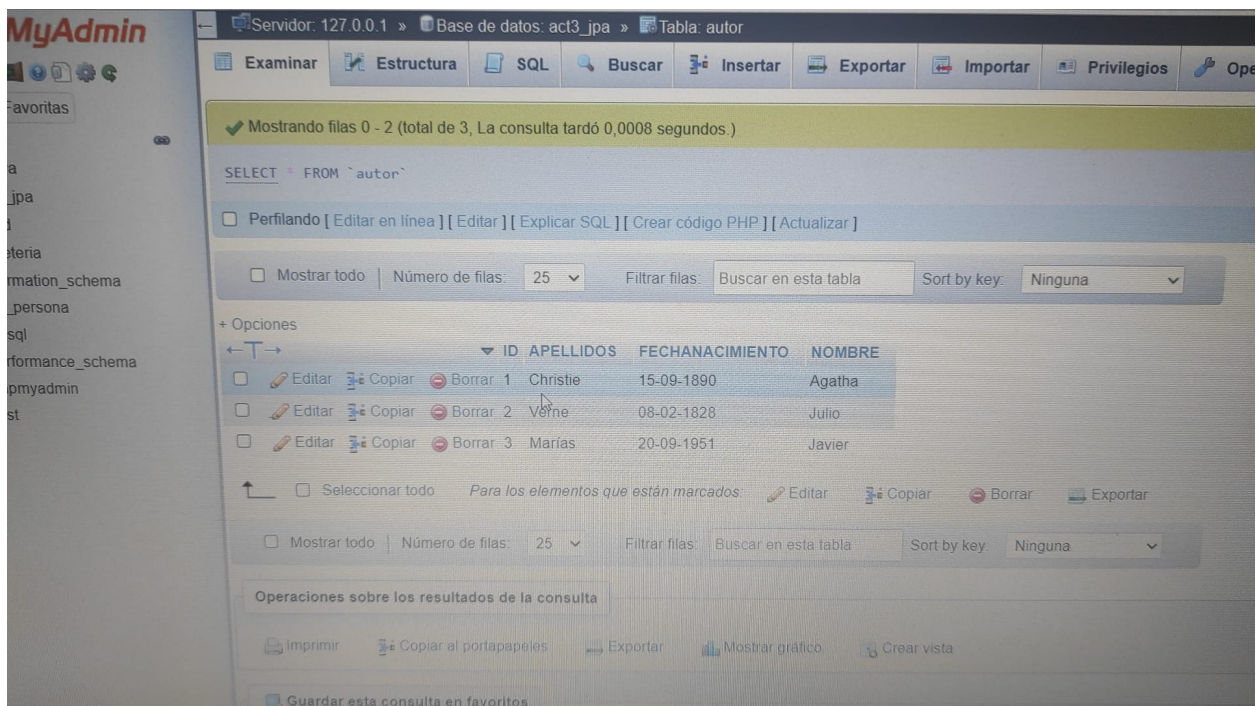
List<Libreria> libreriasLibro7 = new ArrayList<Libreria>();
libreriasLibro7.add(libreria1);
libreriasLibro7.add(libreria2);
libro7.setLibrerias(libreriasLibro7);

```

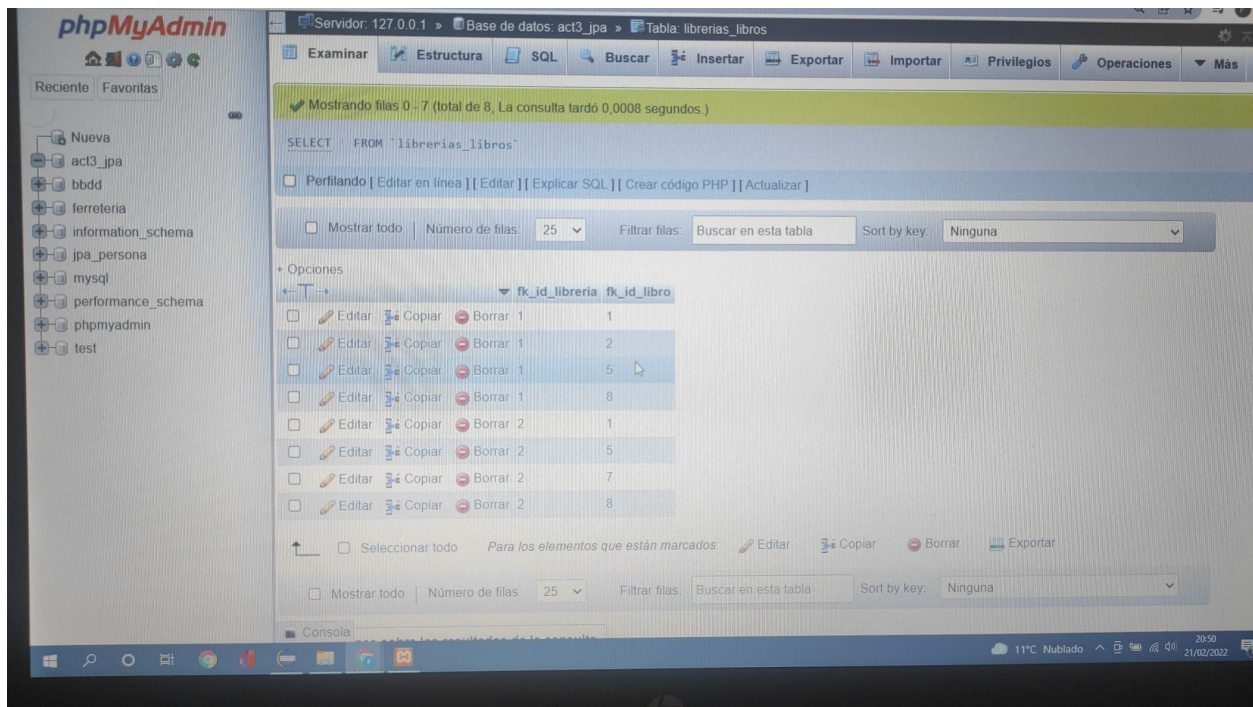
Las diferentes tablas se crean con sus respectivos datos.



Ejemplo con la tabla autor y sus respectivos datos:



La tabla renacida librerias_libros que ha surgido de la entidad N-N:



Requerimiento 2:

Se pide realizar las siguientes consultas y mostrarlas por pantalla, cada una debe de ser independiente:

1. Mostrar todos los libros dados de alta, con su editorial y su autor

```
System.out.println("=====QUERY 1=====");
Query query = em.createQuery("SELECT DISTINCT lib.titulo, lib.editorial, lib.autor FROM Libro lib");
List<Object[]> resultados = query.getResultList();
System.out.println("===Mostrar todos los libros dados de alta, con su editorial y su autor===");
for (Object[] p : resultados) {
    System.out.println(p[0] + " - " + p[1] + " - " + p[2]); // la posicion 0 tiene el nombre y la 1 el telefono
}
```

2. Mostrar todos los autores dados de alta, con sus libros asociados

```
System.out.println("=====QUERY 2=====");
query = em.createQuery("SELECT DISTINCT lib.autor, lib.titulo FROM Libro lib");
List<Object[]> resultados1 = query.getResultList();
System.out.println("===Mostrar todos los autores dados de alta, con sus libros asociados===");
for (Object[] p : resultados1) {
    System.out.println(p[0] + " - " + p[1]);
}
```

3. Mostrar todas las librerías, con solamente sus libros asociados

```
/** NO HEMOS CONSEGUIDO IMPLEMENTARLA
System.out.println("=====QUERY 3=====");
query = em.createQuery("SELECT libr.nombre, libr.coleccionLibros FROM Libreria libr");
System.out.println("===Mostrar todas las librerías, con solamente sus libros asociados===");
List<Object[]> resultados2 = query.getResultList();
for (Object[] p : resultados2) {
    System.out.println(p[0] + " - " + p[1]);
}

**/
```


4. Mostrar todos los libros dados de alta, y en la librería en la que están.

```
/**NO HEMOS CONSEGUIDO IMPLEMENTARLA
System.out.println("=====QUERY 4=====");
query = em.createQuery("SELECT DISTINCT lib.titulo, lib.librerias FROM Libro lib");
System.out.println("===Mostrar todos los libros dados de alta, y en la librería en la que están===");
List<Object[]> resultados2 = query.getResultList();
for (Object[] p : resultados2) {
    System.out.println(p[0] + " - " + p[1]);
}
**/
```

Resultado en consola:

```
=====QUERY 4=====
=====QUERY 1=====
Mostrar todos los libros dados de alta, con su editorial y su autor=====
La isla misteriosa - Editorial [Id=2, nombre=delasia, direccion=avda Madrid, 44, Vigo, librosPublicados[IndirectList: not instantiated]] - Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828]
Los dominios del lobo - Editorial [Id=2, nombre=delasia, direccion=avda Madrid, 44, Vigo, librosPublicados[IndirectList: not instantiated]] - Autor [Id=1, nombre=Javier, apellidos=Marías, fechaNacimiento=28-09-1951]
Misterio en el Caribe - Editorial [Id=1, nombre=Espasa, direccion=Calle Josefa Valcárcel, 42, Madrid, librosPublicados[IndirectList: not instantiated]] - Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890]
Muerte sobre el Nilo - Editorial [Id=2, nombre=delasia, direccion=avda Madrid, 44, Vigo, librosPublicados[IndirectList: not instantiated]] - Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890]
Viaje al centro de la tierra - Editorial [Id=2, nombre=delasia, direccion=avda Madrid, 44, Vigo, librosPublicados[IndirectList: not instantiated]] - Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828]
De la tierra a la luna - Editorial [Id=2, nombre=Espasa, direccion=Calle Josefa Valcárcel, 42, Madrid, librosPublicados[IndirectList: not instantiated]] - Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828]
El suceso del tiempo - Editorial [Id=1, nombre=Espasa, direccion=Calle Josefa Valcárcel, 42, Madrid, librosPublicados[IndirectList: not instantiated]] - Autor [Id=1, nombre=Javier, apellidos=Marías, fechaNacimiento=28-09-1951]
Asesinato en el Orient Express - Editorial [Id=1, nombre=Espasa, direccion=Calle Josefa Valcárcel, 42, Madrid, librosPublicados[IndirectList: not instantiated]] - Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890]
=====QUERY 2=====
Mostrar todos los autores dados de alta, con sus libros asociados=====
Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828] - La isla misteriosa
Autor [Id=1, nombre=Javier, apellidos=Marías, fechaNacimiento=28-09-1951] - Los dominios del lobo
Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890] - Misterio en el Caribe
Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890] - Muerte sobre el Nilo
Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828] - Viaje al centro de la tierra
Autor [Id=2, nombre=Julio, apellidos=Verne, fechaNacimiento=88-02-1828] - De la tierra a la luna
Autor [Id=1, nombre=Javier, apellidos=Marías, fechaNacimiento=28-09-1951] - El suceso del tiempo
Autor [Id=1, nombre=Agatha, apellidos=Christie, fechaNacimiento=15-09-1890] - Asesinato en el Orient Express
```

Requerimiento 3:

Se pide realizar un modelo de datos de la actividad. Hemos realizado un modelo entidad-relación que recoge cómo hemos enfocado nuestra aplicación. El original lo hemos incluido en el proyecto, en la carpeta dibujo:

