**Actividad: Análisis y Desarrollo - Comprendiendo JPA y las Relaciones entre Entidades Objetivo:**

Leer el documento técnico adjunto sobre JPA (Java Persistence API) y sus relaciones entre entidades, para luego desarrollar un pequeño modelo de base de datos aplicado a un caso real.

**Material de lectura:**

Documento sugerido: **“Guía Básica de JPA y mapeo de relaciones”**.

**Instrucciones:**

# Parte 1 - Lectura comprensiva (30 minutos)

* Leer el documento completo.
* Subrayar o resaltar conceptos clave:
  + ¿Qué es JPA? o ¿Cómo se define una entidad? o Tipos de relaciones: OneToOne, OneToMany, ManyToOne, ManyToMany.
  + Uso de @JoinColumn, mappedBy, y CascadeType.
* Realizar un glosario con al menos 10 términos nuevos aprendidos en la lectura, explicados con tus propias palabras.

# Parte 2 - Desarrollo aplicado (60 minutos)

Con base en la lectura, desarrolla un **modelo de entidades** para un sistema sencillo de biblioteca, que debe incluir:

* Entidad **Libro**, relacionada con:
  + Un autor (@ManyToOne)
  + Varios ejemplares físicos (@OneToMany) o Varias categorías (@ManyToMany)
* Entidad **Usuario**, relacionada con: o Préstamos (@OneToMany)
* Entidad **Préstamo**, relacionada con:
  + Un solo ejemplar (@ManyToOne) o Un solo usuario (@ManyToOne) El desarrollo debe incluir:
* Creación de las clases de entidad.
* Aplicación correcta de las anotaciones JPA.
* Diagrama relacional (puede ser a mano o digital).
* Documento breve explicando por qué elegiste cada tipo de relación.

**Producto esperado:**

* Glosario de términos.
* Código fuente de las entidades.
* Diagrama de relaciones entre entidades.
* Documento explicativo (máximo una página).

**DOCUMENTO DE LECTURA: INTRODUCCIÓN A JPA Y RELACIONES ENTRE ENTIDADES**

**¿Qué es JPA?**

La **Java Persistence API (JPA)** es una especificación de Java que define cómo interactuar con bases de datos relacionales desde aplicaciones Java. JPA no es un framework en sí mismo, sino una serie de reglas y anotaciones que frameworks como Hibernate, EclipseLink o OpenJPA implementan.

Con JPA, los desarrolladores pueden mapear clases Java a tablas en una base de datos, permitiendo que los objetos se guarden, consulten, actualicen o eliminen fácilmente.

**¿Qué es una Entidad?**

En JPA, una **entidad** es una clase Java que representa una tabla de base de datos. Cada instancia de la clase representa una fila de esa tabla. Para que una clase sea considerada entidad, debe llevar la anotación:

@Entity

public class Libro {

@Id private Long id; private String titulo;

}

# Tipos de Relaciones en JPA

Las relaciones permiten conectar entidades entre sí, simulando las asociaciones entre tablas en la base de datos. JPA define **cuatro tipos principales de relaciones**:

# OneToOne (Uno a Uno)

Ejemplo: Un proyecto tiene un único evaluador asignado.

@OneToOne

@JoinColumn(name = "evaluador\_id") private Evaluador evaluador;

# OneToMany (Uno a Muchos)

Ejemplo: Un autor puede tener varios libros publicados.

@OneToMany(mappedBy = "autor")

private List<Libro> libros;

# ManyToOne (Muchos a Uno)

Ejemplo: Muchos libros pueden pertenecer a una misma categoría.

@ManyToOne

@JoinColumn(name = "categoria\_id") private Categoria categoria;

# ManyToMany (Muchos a Muchos)

Ejemplo: Un libro puede pertenecer a varias categorías y una categoría puede tener varios libros.

@ManyToMany

@JoinTable(

name = "libro\_categoria", joinColumns = @JoinColumn(name = "libro\_id"), inverseJoinColumns = @JoinColumn(name = "categoria\_id")

)

private List<Categoria> categorias;

# Anotaciones clave de JPA

|  |  |
| --- | --- |
| **Anotación** | **Función** |
| @Entity | Define una clase como entidad. |
| @Table | Define el nombre de la tabla. |
| @Id | Define la clave primaria. |
| **Anotación** | **Función** |
| @GeneratedValue | Configura la estrategia de generación de la clave primaria. |
| @Column | Configura propiedades de una columna específica. |
| @OneToOne | Define una relación uno a uno. |
| @OneToMany | Define una relación uno a muchos. |
| @ManyToOne | Define una relación muchos a uno. |
| @ManyToMany | Define una relación muchos a muchos. |
| @JoinColumn | Define la columna que actúa como clave foránea. |
| @JoinTable | Define una tabla intermedia para relaciones muchos a muchos. |

# Importancia de las Relaciones

En aplicaciones reales, las entidades no existen aisladas. Los sistemas de información funcionan con datos relacionados, por ejemplo:

* Un cliente realiza varios pedidos.
* Un pedido incluye varios productos.
* Cada producto pertenece a una categoría.

Estas relaciones deben mapearse correctamente para que la base de datos refleje correctamente el modelo de negocio.

# Validación de Relaciones

Al definir relaciones, es importante: Configurar correctamente las **claves foráneas** (JoinColumn).

* Definir el lado dueño y el lado inverso (mappedBy).
* Manejar correctamente el **cascading** para propagación de operaciones (persist, merge, remove).
* Configurar la **carga (fetch)**: Lazy o Eager, según el contexto.

# Ejemplo completo: Entidad Libro y su relación con Autor y Categoría

@Entity

public class Libro {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String titulo;

@ManyToOne

@JoinColumn(name = "autor\_id")

private Autor autor;

@ManyToMany

@JoinTable( name = "libro\_categoria", joinColumns = @JoinColumn(name = "libro\_id"), inverseJoinColumns = @JoinColumn(name = "categoria\_id")

)

private List<Categoria> categorias;

}

# Glosario sugerido (completa tú mismo tras leer)

|  |  |
| --- | --- |
| **Término** | **Definición (según lo entendido)** |
| Entidad | En una clase en java que representa una tabla de bases de datos. |
| Relación | Es una conexión que permite conectar entidades entre ellas. |
| **Término** | **Definición (según lo entendido)** |
| Clave primaria | Es un atributo de la entidad que distingue de manera única cada fila. |
| Clave foránea | Es un vínculo entre entidades donde solo se asigna un único registro. |
| OneToOne | Es una relación entre entidades con un único registro asignado. |
| OneToMany | Es un vínculo entre entidades en el que una de ellas está asociada a múltiples registros. |
| JoinColumn | Anotación que especifica en qué columna se guarda la relación con otra tabla. |
| Cascading | Hace que las modificaciones en una entidad impacten también en sus elementos relacionados. |
| Fetch | Especifica si la información vinculada se obtiene de forma inmediata o solo cuando se requiere. |
| Persistencia | Almacenar información en la base de datos para que permanezca disponible incluso después de cerrar la aplicación |