Universidad de Cundinamarca

# Documentación de relaciones entre modelos, etiquetas de las clases y etiquetas de Lombok.

Miguel Angel Bejarano Rojas

01 de abril de 2025

## Etiquetas para relaciones entre modelos

- @OneToOne: Esta etiqueta se utiliza como referencia de una relación uno a uno, esta etiqueta se debe usar en los dos modelos donde se utiliza, debe existir una variable que se relacione con el otro modelo. Esto indica que es una relación de uno a uno, además en el modelo que almacena la llave foránea debe tener una identificación de la columna que se usa como llave foránea, para eso se usa la etiqueta @JoinColumn, de forma que podemos definir el nombre de la columna.
- @OneToMany y @ManyToOne: Esta etiqueta se utiliza como referencia de una relación uno a muchos, en este caso hay que tener en cuenta en qué lugar va cada etiqueta, puesto que en una relación de este tipo donde está la llave foránea es el que hace referencia a uno solo almacenando la llave foránea. Igual que en el caso de @OneToOne se usa el @JoinColumn para definir la llave primaria, pero además dentro de la etiqueta OneToMany se hace referencia a el parámetro que está almacenando la llave foránea.
- @ManyToMany: Esta etiqueta se usa en relaciones muchos a muchos, su funcionamiento es similar a los demás casos, pero en este caso puesto que es de muchos a muchos en el modelo principal se hace la referencia en doble vía tanto hacia el modelo objetivo como desde el modelo objetivo hacia el modelo inicial, esto se logra utilizando @JoinColumn y @InverseJoinColumn para poder nombrar las columnas que se crean siguiente nuestra propia convención.
- @JoinColumn: Su uso permite hacer referencia a el modelo objetivo y darle un nombre a la columna que se creará.
- @InverseJoinColumn: Su uso permite hacer referencia a el modelo en el que se trabaja de forma que se configura el nombre de la columna que tendrá la referencia de la llave foránea tomada desde el modelo actual en el objetivo.

## Etiquetas de clases

- @Entity: permite definir una clase que se convertirá en una tabla en la base de datos.
- @Table: permite definir el nombre de la tabla si se desea modificar, de no ser utilizado se usa el nombre de la clase separado por “_”
- @Id: identifica la variable que representa la llave primaria en la tabla.
- @GeneratedValue: Permite definir como se autogenera el valor de la variable.
- @Column: Esta etiqueta permite controlar las especificaciones de cada campo. ejemplos:
    - @Column(nullable = false)
    - @Column(unique = false)

## Etiquetas de Lombok

- @Getter: crea los getter de cada propiedad de la clase.
- @Setter: crea los setter de cada propiedad de la clase.
- @AllArgsConstructor: crea un constructor con todos los argumentos de la clase.
- @NoArgsConstructor: Crea un constructor sin argumentos.

## Creación de relaciones

**Uno a uno**

Relación uno a uno entre tabla profesor y detalleProfesor.

```
@OneToOne
@JoinColumn(name = "DetalleProfesor")
private DetalleProfesor detalleProfesor;
```

Se crea la etiqueta @OneToOne en la clase profesor y se define el nombre que tendrá la columna que tendrá la llave foránea.

```
@OneToOne(mappedBy = "detalleProfesor")
private Profesor profesor;
```

Tras esto se va a clase DetalleProfesor, se incluye la etiqueta @OneToOne y se le especifica el campo que está definiendo la relación con mappedBy.

**Uno a muchos**

Relación uno a muchos entre Profesor y Estudiante.

```
@ManyToOne
@JoinColumn(name="profesor")
private Profesor profesor;
```

Se define el campo profesor, es importante tener en cuenta que esto se está haciendo en el modelo Estudiante, por lo que la etiqueta requerida en este caso es @ManyToOne y se le asigna el nombre a la columna.

```
@OneToMany(mappedBy = "profesor")
private List<Estudiante> estudiantes;
```

Tras esto se dirige a la definición en la clase profesor y se define el campo de la relación.

**Muchos a muchos**

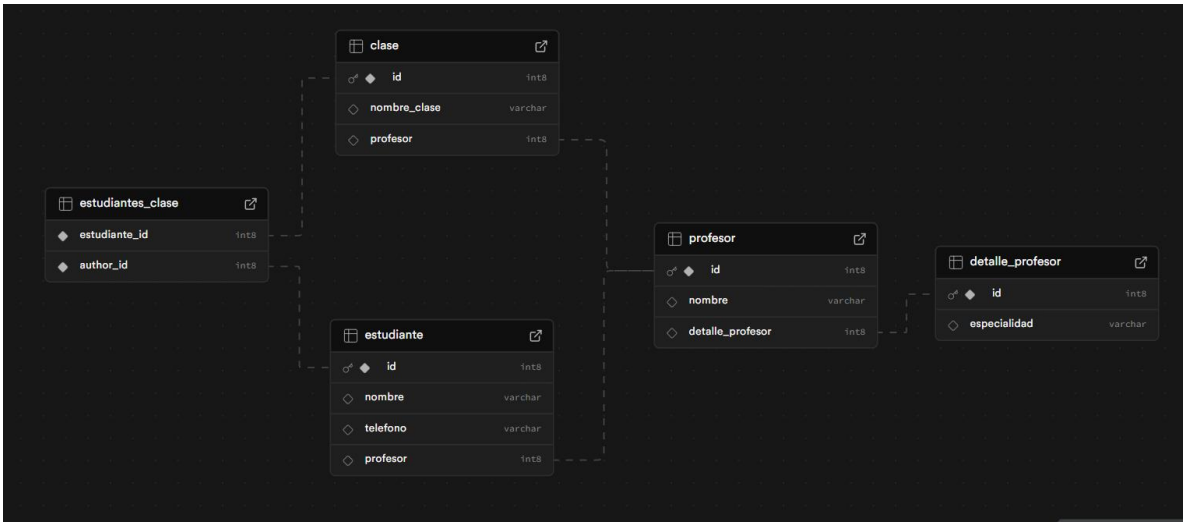
Relación de muchos a muchos entre Estudiante y Clase

```
@ManyToMany(cascade = CascadeType.ALL)
@JoinTable(name = "estudiantes_clase",
    joinColumns = @JoinColumn(name = "estudiante_id", referencedColumnName = "id"),
    inverseJoinColumns = @JoinColumn(name = "author_id", referencedColumnName = "id"))
private List<Estudiante> estudiantes;
```

Se define la etiqueta @ManyToMany en el campo estudiantes, y se definen los nombres de las columnas, además de hacer referencia a la columna que tiene la llave primaria en cada modelo, para hacer la relación.

```
@ManyToMany(mappedBy = "estudiantes")
private List<Clase> clases;
```

Puesto que la relación ya se hizo desde Clase, no es necesario hacerlo en Estudiante, solo hay que hacer referencia a el campo que tiene la referencia.




Link del repositorio

https://github.com/MignoDev/DBRemoto

**Referencias**

StackOverflow. (2021) https://stackoverflow.com/questions/65059299/how-can-i-set-the-not-null-field-spring-boot-hibernate

Fejér Attila & Martin(Mayo 11, 2024). *EricMany-To-Many Relationship in JPA*. https://www.baeldung.com/jpa-many-to-many

Mvinntec (Febrero 06, 2020). *Guía de anotaciones de Spring Framework. https://mvitinnovaciontecnologica.wordpress.com/2020/02/06/guia-de-anotaciones-de-spring-framework/*