

ONE TO ONE

TABLAS

Un Castillo tiene un Foso y un Foso pertenece a un Castillo

Nombre:		castillos	
Columnas:		<div>+ Agregar × Borrar</div>	
#	Nombre		
1	id		
2	descripcion		
3	localidad		
4	fecha_construccion		
5	id_foso		

Nombre:		foso	
Columnas:		<div>+ Agregar × B</div>	
#	Nombre		
1	id		
2	profundidad		

ENTIDADES (Clases)

Unidireccional Castillo → Foso (Castillo tiene un Foso):

```
public class Castillo {  
    @OneToOne  
    @JoinColumn(name = "id_foso")  
    private Foso foso;  
}
```

```
public class Foso {  
    // No hay ningún atributo  
    // de tipo Castillo  
}
```

Unidireccional Foso → Castillo (Foso tiene un Castillo):

NO ES POSIBLE ESTA RELACIÓN, porque la columna FK del JOIN está en la tabla CASTILLOS

Bidireccional Castillo ↔ Foso (Castillo tiene un Foso y Foso tiene un Castillo):

```
public class Castillo {  
    @OneToOne  
    @JoinColumn(name = "id_foso")  
    private Foso foso;  
}
```

```
public class Foso {  
    @OneToOne(mappedBy = "foso")  
    private Castillo castillo;  
}
```

ONE TO MANY y MANY TO ONE

TABLAS

Un Castillo tiene muchos Soldados y un Soldado pertenece a un único Castillo

Nombre:	castillos
Columnas:	<div>+ Agregar</div> <div>× Borrar</div>
#	Nombre
1	id
2	descripcion
3	localidad
4	fecha_construccion
5	id_foso

Nombre:	soldado
Columnas:	<div>+ Agregar</div> <div>× Borrar</div>
#	Nombre
1	id
2	nombre
3	rango
4	id_castillo ← FK

ENTIDADES (Clases)

Unidireccional Castillo → Soldado (Castillo tiene una lista de Soldado):

```
public class Castillo {  
    @OneToMany  
    @JoinColumn(name = "id_castillo")  
    private List<Soldado> soldados;  
}
```

```
public class Soldado {  
    // No tiene ningún atributo  
    // de tipo Castillo  
}
```

Unidireccional Soldado → Castillo (Soldado tiene un Castillo):

```
public class Castillo {  
    // No tiene ningún atributo  
    // de tipo Soldado  
}
```

```
public class Soldado {  
    @ManyToOne  
    @JoinColumn(name = "id_castillo")  
    private Castillo castillo;  
}
```

Bidireccional Castillo ↔ Soldado (Castillo tiene una lista de Soldado y Soldado tiene un Castillo):

```
public class Castillo {  
    @OneToMany(mappedBy = "castillo")  
    private List<Soldado> soldados;  
}
```

```
public class Soldado {  
    @ManyToOne  
    @JoinColumn(name = "id_castillo")  
    private Castillo castillo;  
}
```

MANY TO MANY

TABLAS

Un Castillo tiene muchos Mercenarios y un Mercenario pertenece a muchos Castillos

Nombre: castillos

Columnas:

+ Agregar

x Borrar

#	Nombre
1	id
2	descripcion
3	localidad
4	fecha_construccion
5	id_foso

Nombre: mercenario

Columnas:

+ Agregar

x Borrar

#	Nombre
1	id
2	nombre

Nombre: castillo_mercenario

Columnas:

+ Agregar

x Borrar

#	Nombre
1	id_castillo
	id_mercenario

ENTIDADES (Clases)

Unidireccional Castillo → Mercenario (Castillo tiene una lista de Mercenario):

```
public class Castillo {  
    @ManyToMany  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_castillo") },  
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })  
    private List<Mercenario> mercenarios;  
}  
  
public class Mercenario {  
    // No tiene ningún atributo  
    // de tipo Castillo  
}
```

Unidireccional Mercenario → Castillo (Mercenario tiene una lista de Castillo):

```
public class Castillo {  
    // No tiene ningún atributo  
    // de tipo Mercenario  
}  
  
public class Mercenario {  
    @ManyToMany  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_castillo") },  
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })  
    private List<Castillo> castillos;  
}
```

Bidireccional Castillo ↔ Mercenario (Castillo tiene una lista de Mercenario y Mercenario tiene una lista de Castillo):

Hay varias opciones. La más sencilla es poner JoinTable en las dos entidades:

```
public class Castillo {  
    @ManyToMany(cascade = CascadeType.ALL)  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_castillo") },  
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })  
    private List<Mercenario> mercenarios;  
}  
  
public class Mercenario {  
    @ManyToMany(cascade = CascadeType.ALL)  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_mercenario") },  
        inverseJoinColumns = {@JoinColumn(name = "id_castillo") })  
    private List<Castillo> castillos;  
}
```

NOTA: Ten en cuenta que en joinColumns ponemos la FK perteneciente a la entidad donde ponemos el JoinTable, y en inverseJoinColumns ponemos la otra.

Pero esta opción es redundante. Normalmente se pone JoinTable sólo en la entidad principal, y en la otra un mappedBy.

Así, si nuestra entidad principal es Castillo, pondríamos:

```
public class Castillo {  
    @ManyToMany(cascade = CascadeType.ALL)  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_castillo") },  
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })  
    private List<Mercenario> mercenarios;  
}
```

```
public class Mercenario {  
    @ManyToMany(mappedBy = "mercenarios")  
    private List<Castillo> castillos;  
}
```

O al revés:

```
public class Castillo {  
    @ManyToMany(mappedBy = "castillos")  
    private List<Mercenario> mercenarios;  
}
```

```
public class Mercenario {  
    @ManyToMany(cascade = CascadeType.ALL)  
    @JoinTable(name = "castillo_mercenario",  
        joinColumns = {@JoinColumn(name = "id_mercenario") },  
        inverseJoinColumns = {@JoinColumn(name = "id_castillo") })  
    private List<Castillo> castillos;  
}
```