

---

# Herramientas de mapeo objeto-relacional (ORM)

## Proyecto Unidad 3



---

**Autor:**

Antonio Jesús Escudero Godoy

**Titulación:**

Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

**Módulo:**

Acceso a Datos

**Centro de estudios:**

I.E.S. Torre de los Guzmanes

---

## Índice

Sprint 1 Demo Day 02/12/24.....	4
Objetivo de tu sistema.....	5
Modelo de datos y sus relaciones.....	5
Esquema del modelo relacional de la BD.....	7
Script de creación de la BBDD.....	11
Enlace al proyecto en GitHub.....	12

---

## PROYECTO U3. Herramientas de mapeo objeto-relacional (ORM)

### Sprint 1 Demo Day 02/12/24

Desarrolla una aplicación con Hibernate que permita gestionar los datos de una base de datos concreta diferente de la que hemos trabajado en clase.

Se pide REALIZAR y EXPLICAR cada uno de los puntos siguientes:

1. Define el objetivo de tu sistema: qué gestiona en una dos o tres líneas.
2. Define el modelo de datos y sus relaciones. Al menos tienes que tener 6 entidades con sus correspondientes atributos. El sistema deberá contener, como mínimo una relación de cada tipo: 1:1, 1:N unidireccional y otra bidireccional, N:M.
3. Define su Esquema del modelo relacional de la BD escogida.
4. Creación de las clases persistentes y mapeo.
5. Crea una clase Servicio que contenga los DAOS y que tenga una operación del CRUD por cada modelo. Estas acciones serán:
  1. Creación y guardado de objetos en BD.
  2. Borrado de objetos.
  3. Actualización de objetos.
6. Crea una clase Controlador desde donde se utilicen cada una de las operaciones definidas.
7. Añade a tus DAOS y a tu clase servicio las siguientes consultas:
  - 7.1. Una consulta a la BBDD que restrinja el número de elementos devueltos de una lista a 1.
  - 7.2. Dos consultas parametrizadas.

Comentarios:

- No hay que dar las mismas opciones sobre todas las entidades, sino que podéis poner ciertas operaciones básicas sobre algunas de ellas, otras particulares de una entidad concreta, y sobre todo algunas que expliquen el comportamiento de las relaciones entre entidades (por ejemplo: Si borro un departamento, qué ocurre con los empleados que pertenecen a ese departamento y por qué ocurre esto).
- Se valorará la variedad e idoneidad de operaciones para explicar cada tipo de relación.

- 
- Se tendrá también en cuenta el valor añadido, es decir, que se hayan añadido aspectos que no se pedían como puede ser una interfaz gráfica, o el uso de elementos que no hayamos visto en clase sobre Hibernate,... que hagan que la aplicación sea más profesional.

#### ENTREGABLES:

- La entrega se realizará a través de GitHub. Añádeme como colaboradora a tu proyecto (sorayapeceno) y copia la ruta del repositorio. En dicho repositorio debe aparecer:
  - Documento con objetivo, Diagrama que represente el modelo relacional de la BD, diagrama de clases, Modelo de datos. Debe estar en la carpeta resources.
  - Proyecto con Javadoc generado.
  - Script de creación de la BBDD.

### Objetivo de tu sistema

El sistema gestiona la información de un cine, incluyendo la organización de salas, la relación entre el cine y la asociación de la salas con los proyectores, la tienda bar, el catálogo de productos, las compras realizadas por los espectadores y las películas visualizadas.

### Modelo de datos y sus relaciones

#### Entidades y sus atributos:

**Cine** (id, nombre, ubicación)

**TiendaBar** (id, nombre)

**Pedido** (id, importe, fecha)

**Producto** (id, nombre, categoría, precio)

**Espectador** (dni, nombre, edad)

**Película** (id, titulo, genero, duración)

**Sala** (id, nombre, capacidad)

**Proyector** (id, nombre, modelo, marca)

## Relaciones:

### Relación 1:1

**Sala - Proyector:** Una sala tiene un único proyector y un proyector pertenece a una sola sala.

### Relación 1:N unidireccional

**TiendaBar - Producto:** Una tienda bar tiene varios productos, pero cada producto pertenece a una única tienda bar. He considerado que cada tienda tiene productos exclusivos.

**Película - Espectador:** Una película puede ser visualizada por varios espectadores y un espectador visualiza una película.

### Relación 1:N bidireccional

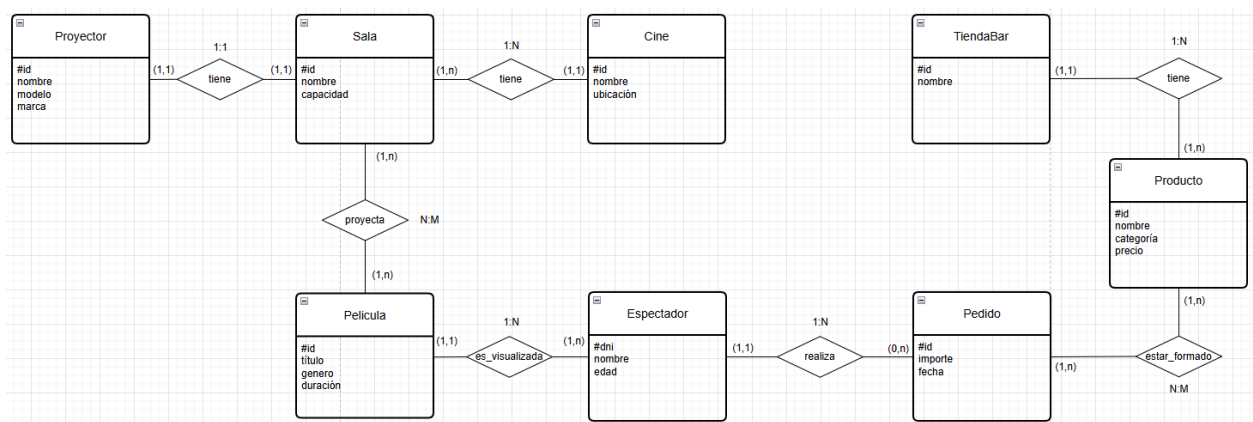
**Espectador - Pedido:** Un espectador puede o no realizar un pedido y un pedido es realizado por un espectador.

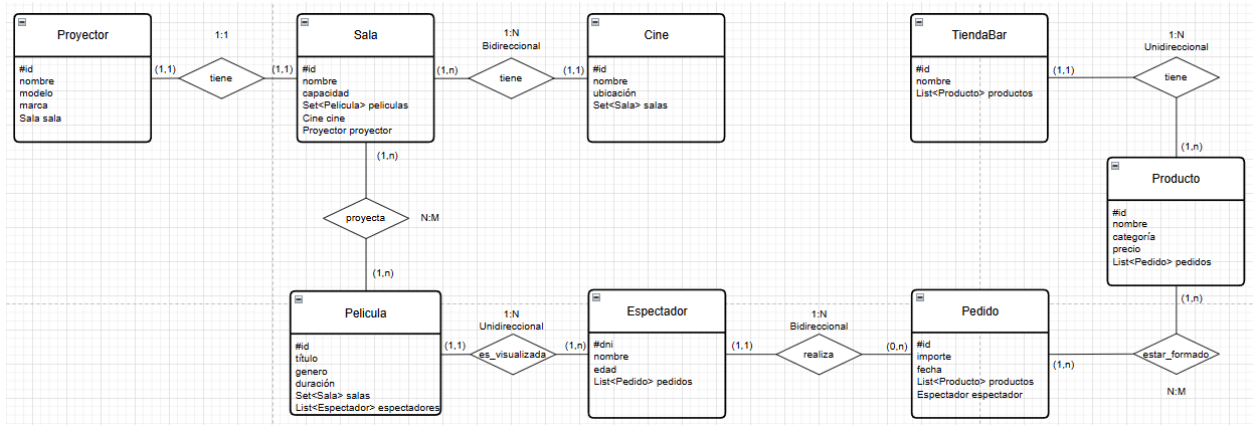
**Cine - Sala:** Un cine puede tener varias salas y una sala solo puede estar en un cine.

### Relación N:M

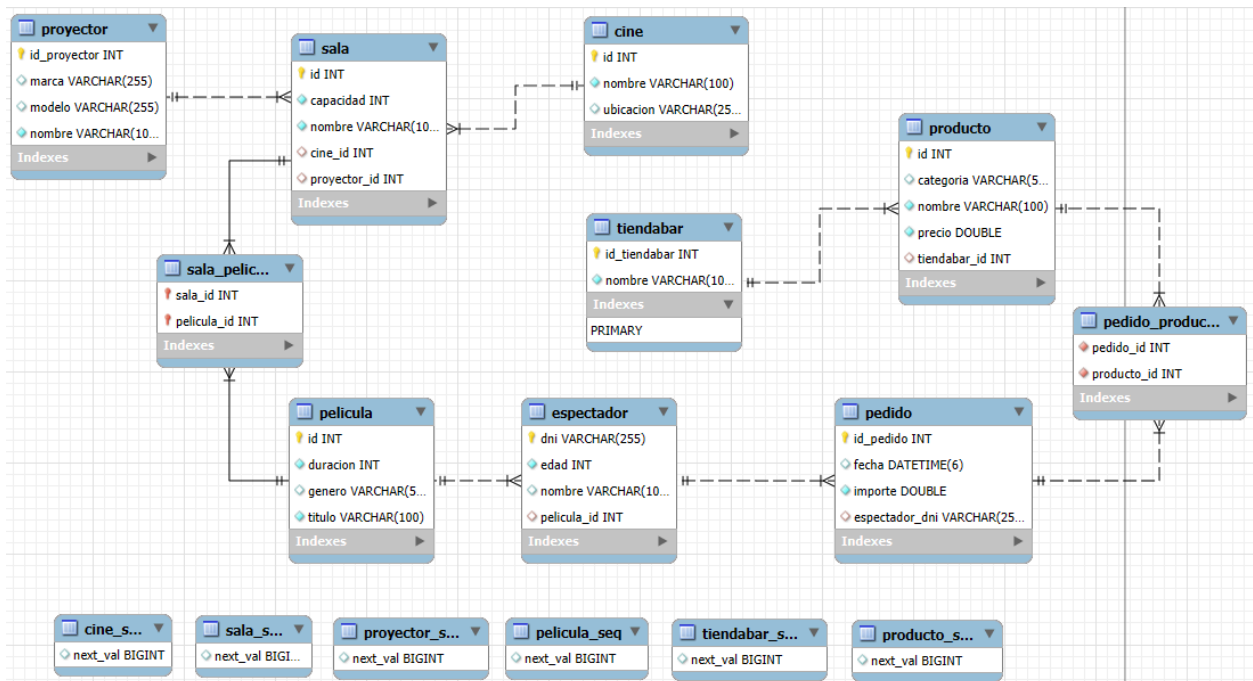
**Sala - Película:** Una sala puede proyectar varias películas en un mismo día a diferentes horas y una película puede ser proyectada en varias salas.

**Producto - Pedido:** Un producto puede estar incluido en muchos pedidos y un pedido puede estar formado de muchos productos.





## Esquema del modelo relacional de la BD



El modelo relacional representa un sistema para la gestión de cines, salas, proyectores, películas, espectadores, pedidos, productos y tiendas.

### Entidades principales

#### 1. película

- Representa las películas que se proyectan en el cine.
- Atributos:
  - id:** Identificador único de la película (clave primaria).

- 
- **duracion:** Duración de la película en minutos.
  - **genero:** Género de la película.
  - **titulo:** Título de la película.
  - Relaciones:
    - **pelicula - espectador:**
      - **Tipo: Uno a muchos (1:N).**
      - **Explicación:** Una película puede tener muchos espectadores que la ven, pero un espectador solo puede ver una película a la vez. Esto se relaciona mediante el campo **pelicula\_id** en la tabla **espectador**.
    - **pelicula - sala (a través de sala\_pelicula):**
      - **Tipo: Muchos a muchos (N:M).**
      - **Explicación:** Una película puede proyectarse en varias salas, y una sala puede proyectar diferentes películas. La relación se gestiona mediante la tabla intermedia **sala\_pelicula**.

## 2. espectador

- Representa a los clientes o espectadores del cine.
- Atributos:
  - **dni:** Identificador único del espectador (clave primaria).
  - **edad:** Edad del espectador.
  - **nombre:** Nombre del espectador.
  - **pelicula\_id:** Identifica la película que el espectador está viendo.
- Relaciones:
  - **espectador - pelicula:**
    - **Tipo: Muchos a uno (N:1).**
    - **Explicación:** Un espectador solo puede ver una película a la vez, pero una película puede tener muchos espectadores.
  - **espectador - pedido:**
    - **Tipo: Uno a muchos (1:N).**
    - **Explicación:** Un espectador puede realizar varios pedidos, pero un pedido está asociado con un único espectador. Esto se gestiona mediante el campo **espectador\_dni** en la tabla **pedido**.

## 3. pedido

- Representa las órdenes realizadas por los espectadores en la tienda bar.
- Atributos:
  - **id\_pedido:** Identificador único del pedido.
  - **fecha:** Fecha y hora en que se realizó el pedido.



- **importe:** Total del pedido.
- **espectador\_dni:** Relación con el espectador que realizó el pedido.
- Relaciones:
  - **pedido - espectador:**
    - **Tipo: Muchos a uno (N:1).**
    - **Explicación:** Cada pedido está asociado con un único espectador, pero un espectador puede realizar varios pedidos.
  - **pedido - producto:**
    - **Tipo: Muchos a muchos (N:M).**
    - **Explicación:** Un pedido puede incluir varios productos un producto puede estar incluido en varios pedido. Esto se gestiona mediante la tabla **pedido\_producto**.

#### 4. **pedido\_producto**

- **Atributos:**
  - **pedido\_id** (clave foránea que apunta a **pedido**).
  - **producto\_id** (clave foránea que apunta a **producto**).
- **Relaciones:**
  - **pedido\_producto - pedido y sala:**
    - **Tipo: Muchos a muchos (N:M).**
    - **Explicación:** Es una tabla intermedia que gestiona la relación muchos a muchos entre **producto** y **sala**.

#### 5. **producto**

- Representa los productos disponibles en la tienda bar (como palomitas, bebidas, etc.).
- **Atributos:**
  - **id:** Identificador único del producto.
  - **categoria:** Categoría del producto (por ejemplo, comida, bebida).
  - **nombre:** Nombre del producto.
  - **precio:** Precio del producto.
  - **tiendabar\_id:** Identificador de la tienda bar que vende el producto.
  - **pedido\_id:** Identificador del pedido asociado al producto.
- **Relaciones:**
  - **producto - pedido:**
    - **Tipo: Muchos a muchos (N:M).**
    - **Explicación:** Un producto puede pertenecer a múltiples pedidos y un pedido puede contener múltiples productos.
  - **producto - tiendabar:**
    - **Tipo: Muchos a uno (N:1).**

- **Explicación:** Cada producto es ofrecido por una única tienda bar, pero una tienda puede ofrecer muchos productos. Esto se gestiona mediante el campo `tiendabar_id`.

## 6. sala

- Representa las salas donde se proyectan las películas.
- Atributos:
  - `id`: Identificador único de la sala.
  - `capacidad`: Capacidad de la sala.
  - `nombre`: Nombre de la sala.
  - `cine_id`: Identificador del cine al que pertenece la sala.
- Relaciones:
  - **sala - cine:**
    - **Tipo: Muchos a uno (N:1).**
    - **Explicación:** Una sala pertenece a un único cine, pero un cine puede tener varias salas.
  - **sala - pelicula (a través de sala\_pelicula):**
    - **Tipo: Muchos a muchos (M:N).**
    - **Explicación:** Una sala puede proyectar múltiples películas en diferentes días, y una película puede proyectarse en varias salas. Esto se gestiona mediante la tabla `sala_pelicula`.

## 7. sala\_pelicula

- **Atributos:**
  - `sala_id` (clave foránea que apunta a `sala`).
  - `pelicula_id` (clave foránea que apunta a `pelicula`).
- **Relaciones:**
  - **sala\_pelicula - sala y pelicula:**
    - **Tipo: Muchos a muchos (N:M).**
    - **Explicación:** Es una tabla intermedia que gestiona la relación muchos a muchos entre `sala` y `pelicula`.

## 8. cine

- Representa los cines.
- Atributos:
  - `id`: Identificador único del cine.
  - `nombre`: Nombre del cine.
  - `ubicacion`: Ubicación del cine.
- Relaciones:
  - **cine - sala:**
    - **Tipo: Uno a muchos (1:N).**

- **Explicación:** Un cine puede tener varias salas, pero una sala pertenece a un único cine.

## 9. tiendabar

- Representa las tiendas donde se venden productos exclusivos.
- Atributos:
  - **id\_tiendabar:** Identificador único de la tienda.
  - **nombre:** Nombre de la tienda.
- Relaciones:
  - **tiendabar - producto:**
    - **Tipo: Uno a muchos (1:N).**
    - **Explicación:** Una tienda puede ofrecer múltiples productos, pero cada producto pertenece a una única tienda.

## 10. proyector

- Representa los proyectores de las salas de cine.
- Atributos:
  - **id\_proyector:** Identificador único del proyector.
  - **nombre:** Nombre del proyecto.
  - **modelo:** Modelo del proyecto.
  - **marca:** Marca de proyecto.
- Relaciones:
  - **proyector - sala:**
    - **Tipo: Uno a uno (1:1).**
    - **Explicación:** Cada proyector pertenece a una única sala, y una sala tiene solo un proyector.

## Entidades auxiliares (secuencias)

Las entidades con **\_seq** se utilizan para gestionar las secuencias de identificadores en las tablas principales (ID autogenerados):

- **pedido\_seq:** Secuencia para generar IDs únicos para los pedidos.
- **producto\_seq:** Secuencia para los productos.
- **sala\_seq:** Secuencia para las salas.
- **cine\_seq:** Secuencia para los cines.
- **tiendabar\_seq:** Secuencia para las tiendas.
- **proyector\_seq:** Secuencia para los proyectores.

## Script de creación de la BBDD

-- Crear usuario

---

```
CREATE USER 'godoy'@'localhost' IDENTIFIED BY '4321';
```

**-- Dar privilegios específicos**

```
GRANT ALL PRIVILEGES ON cinebasedatos.* TO 'godoy'@'localhost';
```

**-- Verificar los privilegios**

```
SHOW GRANTS FOR 'godoy'@'localhost';
```

**-- Eliminar base de datos existente**

```
DROP DATABASE IF EXISTS cinebasedatos;
```

**-- Crear base de datos**

```
CREATE DATABASE cinebasedatos;
```

**-- Usar base de datos**

```
USE cinebasedatos;
```

## Enlace al proyecto en GitHub

[https://github.com/ajesgodoy/proyecto\\_hibernate\\_ajeg.git](https://github.com/ajesgodoy/proyecto_hibernate_ajeg.git)