

**Acceso a datos**

**Actividad UT 05 - 01**

**Uso de JPA con Hibernate para acceder a MySql / MaríaDB**

**CONTENIDO**

[1.- Objetivos 2](#_Toc183690450)

[2.- Visión general de la actividad 2](#_Toc183690451)

[3.- Equipos de trabajo 2](#_Toc183690452)

[4.- Requisitos previos 2](#_Toc183690453)

[5.- Entregas y evaluación 2](#_Toc183690454)

[6.- Actividades a realizar 2](#_Toc183690455)

[6.1.- Mapeo de entidades 2](#_Toc183690456)

[6.2.- Creación de propiedades de navegación 3](#_Toc183690457)

[6.3.- Programas de prueba. 4](#_Toc183690458)

# 1.- Objetivos

* Familiarizarse con el uso de herramienta ORM en Java
* Familiarizarse con las herramientas ORM integradas en IntelliJ Idea
* Practicar el desarrollo de modelos de clases siguiendo un enfoque “database first”
* Practicar la creación de relaciones entre clases, y la navegación a través de estas relaciones

# 2.- Visión general de la actividad

La actividad está enmarcada dentro de la UT 05 (Herramientas de mapeo objeto-relacional). En esta actividad se creará un modelo de clases, mapeado con JPA y el ORM Hibernate, para luego realizar una serie de programas que permitan probar este modelo / mapeo.

# 3.- Equipos de trabajo

Esta actividad se realizará de forma individual. No hay equipos de trabajo

# 4.- Requisitos previos

* Instancia de MySQL o MariaDB con la base de datos de demo “Sakila” configurada, incluido un usuario específico para acceder al esquema / base de datos “sakila”. Cada alumno debe disponer de su propia instancia de la BD.
* IntelliJ Idea. Se podrá usar cualquier otro entorno, pero las explicaciones y los ejemplos de clase se basan en este entorno. No se recomienda Eclipse, a no ser que sea la versión para Java EE. NetBeans tiene herramientas de ayuda al ORM similares a las de IntelliJ.

# 5.- Entregas y evaluación

Se entregará el proyecto completo comprimido en zip. En este tipo de proyecto puede haber ficheros fuera de la carpeta SRC necesarios para la correcta ejecución (ej:pom.xml)

# 6.- Actividades a realizar

## 6.1.- Mapeo de entidades

Crear clases (entidades) para las siguientes tablas de la BD Sakila:

* Actor
* Address
* Category
* City
* County
* Customer
* Film
* Inventory
* Language
* Payment
* Rental
* Staff
* Store

El resto de las tablas / vistas no deben crearse ni mapearse. Se utilizarán en algún caso para establecer relaciones y propiedades de navegación

## 6.2.- Creación de propiedades de navegación

En las entidades previamente creadas, crear las siguientes propiedades de navegación:

* Actor
  + Colección de películas en las que participa
* Address
  + Ciudad de la dirección
* Category
  + Sin propiedades de navegación
* City
  + El país al que pertenece
* County
  + Sin propiedades de navegación
* Customer
  + La dirección del cliente
  + La tienda a la que está asociado
  + Los alquileres que ha realizado
  + Los pagos que ha realizado
* Film
  + Colección de categorías a las que pertenece
  + El idioma original de la película
  + El idioma de la película
* Inventory
  + La película
  + La tienda en la que está la copia de la película
* Language
  + Sin propiedades de navegación
* Payment
  + Sin propiedades de navegación
* Rental
  + El empleado que realizó el alquiler
  + El elemento de inventario al que está asociado el alquiler
  + Los pagos realizados asociados al alquiler
  + El cliente que realizó el alquiler
* Staff
  + La tienda a la que está asociado el empleado
  + La dirección del empleado
  + Los alquileres que ha realizado
* Store
  + Los empleados que pertenecen a una tienda

A tener en cuenta:

* Las relaciones pueden ser OneToMany, ManyToOne, ManyYoMany, etc. en función de la cardinalidad, y de si hay tablas intermedias no mapeadas, que sirven para unir tablas en relaciones N a M.
* A veces, al añadir relaciones @ManyToOne a las entidades, algunos de sus atributos se hacen redundantes, porque lo tenemos como parte de la entidad (clave ajena), pero también como parte de la relación (id del elemento relacionado). En estos casos hay dos opciones:
  + Eliminar el atributo si no lo consideramos necesario. Es lo más correcto desde el enfoque de la orientación a objetos, porque el atributo que es clave ajena, forma parte de la entidad relacionada.
  + Decir al ORM que no se va a modificar este atributo, con los parámetros updateable e insertable de la anotación @Column. Si se produce este problema el ORM suele sugerir este cambio. Esta solución no es la más correcta desde el punto de vista de la orientación a objetos.

## 6.3.- Programas de prueba.

Crear una serie de programas de prueba para comprobar que el mapeo funciona. En esta actividad no se puede usar Spring Data. Esto quiere decir que no habrá repositorios. Se tendrá que usar un EntityManager para acceder a la BD. Consultar los ejemplos en el repositorio como referencia.

Cada programa será una clase con un método main, y, si se considera necesario, métodos auxiliares.

En todos los casos en los que haya que buscar (buscar cliente, buscar película, etc.), si no se encuentra el elemento buscado, se mostrará un mensaje de información, no se pueden lanzar excepciones de tipo NullPointerException o cualquier otra relacionadas con el acceso a las entidades. Los errores deben ser controlados.

No hace falta control de entrada de usuario. Si el usuario introduce un valor no numérico para un número el programa puede fallar, no se penalizará por estas excepciones.

Cuando se indica que se muestran los datos de una entidad, quiere decir que se mostrarán TODOS los datos de la entidad, salvo las relaciones. Ejemplo, para el cliente, se mostrarían id, nombre, apellido, email, si está activo o no, fecha de creación y fecha de modificación.

Las relaciones (propiedades de navegación) que se deseen mostrar se indicarán en cada programa. Los datos de las propiedades de navegación se deben extraer a través de la propiedad de navegación. No se pueden hacer consultas adicionales al EntityManager para obtener los datos de las entidades relacionales. El EntityManager realizará las consultas necesarias de forma transparente.

### Informe de situación de cliente.

* Pide al usuario el ID de un cliente
* Muestra para el cliente:
  + Los datos del cliente.
  + Su dirección, incluida ciudad y provincia.
  + La tienda a la que está asociado, con la dirección de esta, incluida ciudad y provincia.
  + Los alquileres que ha realizado, incluido el título de la película.
  + Los pagos que ha realizado
* Vuelve a preguntar al usuario un id. Cuando introduzca un cero, el programa termina.

### Informe de película

* Pide al usuario el id de una película.
* Muestra para la película:
  + Los datos de la película.
  + Las categorías a las que pertenece
  + El idioma original de la película
  + El idioma de la película
  + Los actores que participan en la película.
  + Las copias disponibles de la película en el inventario, incluyendo la tienda y la dirección de la tienda en la que está la copia, incluyendo ciudad y provincia.
* Vuelve a preguntar al usuario un id. Cuando introduzca cero, el programa termina.

### Informe de empleado

* Pide al usuario el id de un empleado.
* Para el empleado, muestra:
  + Los datos del empleado
  + La dirección del empleado, con ciudad y provincia
  + La tienda a la que está asociado, incluida su dirección, con ciudad y provincia
  + Los alquileres que ha realizado, incluyendo el título de la película
* Vuelve a preguntar al usuario un id. Cuando introduzca un cero, el programa termina.

### Alquiler de película

* Pide al usuario los siguientes datos:
  + Id del empleado que está realizando el alquiler
    - Del id del empleado se extraerá el id de la tienda en la que está la película.
  + Id de la película que se quiere alquilar
  + Id del cliente que está haciendo el alquiler
* Realiza las siguientes tareas:
  + Comprueba que el empleado existe. Si no existe, informa del error y termina.
  + Obtiene de este empleado la tienda en la que debe buscar la película.
  + Comprueba que la película existe en el inventario de la tienda. Si no existe, informa del error y termina.
  + Comprueba que haya copias disponibles. Si no las hay informa al usuario y el programa termina.
    - Nota: esto se puede hacer de varias formas. Si se desea, se pueden usar los procedimientos almacenados disponibles en la BD Sakila para comprobar el stock. Esta parte implica algo de investigación si se desea usar procedimientos almacenados.
  + Comprueba que el cliente exista. Si no existe, informa del error y termina.
  + Si todas las comprobaciones previas han funcionados, creará un nuevo alquiler (rental), asociando al alquiler las entidades relacionadas.