

## Contenido

Objetivos de la actividad formativa.....	3
Estructuración y metodología de la actividad formativa.....	3
Requisitos para ejecutar la aplicación de training.....	4
Información sobre la aplicación de training.....	12
Contenido de la aplicación. ....	12
Arquitectura. ....	14
Parte I – Acceso a datos con Java. ....	15
Librería jdbc de MySQL. ....	15
Añadir la librería jdbc de MySQL a un proyecto en NetBeans. ....	16
Interfaces Java necesarias para conectar por jdbc a una base de datos.....	21
Conexión a base de datos. Interfaz <i>Connection</i> . ....	21
Ejecución de consultas. Interfaces <i>PreparedStatement</i> y <i>ResultSet</i> . ....	24
Actualización de registros. ....	27
Eliminación e inserción de registros.....	28
Parte II – Interfaz de usuario con Java.....	29

## Ilustraciones

Ilustración 1 - Captura de la web de MySQL con la descarga de la base de datos de ejemplo (sakila) .....	5
Ilustración 2 - Contenido del paquete de descarga de la base de datos de ejemplo .....	5
Ilustración 3 - Comprobación de la correcta instalación de la base de datos de ejemplo .....	6
Ilustración 4 - Modelo de la base de datos de ejemplo abierto con el Workbench de MySQL ....	7
Ilustración 5 - Librería de MySQL para Java incluida en la aplicación de training .....	8
Ilustración 6 - Verificación de la correcta instalación de la aplicación de training .....	9
Ilustración 7 - Clase java City y tabla city equivalentes .....	11
Ilustración 8 - Paquetes de la aplicación de training .....	12
Ilustración 9 - Contenido de la aplicación .....	14
Ilustración 10 - Arquitectura por capas de la aplicación .....	15
Ilustración 11 - Librerías de un proyecto NetBeans .....	16
Ilustración 12 – Ver las propiedades de un proyecto en NetBeans .....	17
Ilustración 13 - Configurar librerías de un proyecto .....	18
Ilustración 14 - Añadir una librería externa al classpath .....	19
Ilustración 15 - Selección del driver jdbc de MySQL .....	20
Ilustración 16 - Librería añadida al classpath .....	20
Ilustración 17 - Últimos registros mostrados de la consulta de ciudades .....	24
Ilustración 18 – Películas filtradas por rango de duración .....	26
Ilustración 19 - Ejecución del cambio de descripción de una película .....	27

**IMPORTANTE. Se trata de una actividad formativa autoguiada, lee por ello todo el contenido y evita saltarte partes.**

## Objetivos de la actividad formativa.

En el presente documento, se desarrollan los **contenidos** de las dos últimas unidades temáticas del curso:

- **Acceso a datos** desde aplicaciones Java. Se introducirán y practicarán los conceptos relacionados con el acceso a base de datos desde Java, las clases y métodos utilizados, librerías necesarias para interactuar con la base de datos y buenas prácticas a seguir.
- **Interfaz de usuario** desde aplicaciones Java. Se aprenderá a definir una interfaz de usuario robusta, en detrimento de la e/s estándar que dejaremos de lado, definiendo formularios compuestos de distintos controles visuales y cuyos eventos se programarán para manejar la interacción entre el usuario y la aplicación.

Dicho esto, los **objetivos** a perseguir son:


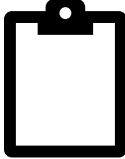
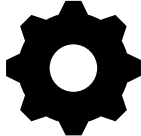
- Evidentemente, **acometer el aprendizaje** de ambas unidades temáticas.
- **Desarrollar la capacidad de autoaprendizaje** del alumno/a, siguiendo una actividad formativa integrada. Esto se conseguirá presentando los contenidos a la par que se ilustran fragmentos de un programa de training que se suministra junto al presente documento. Con ello, el alumnado podrá seguir una formación guiada, ver, editar y ejecutar el código que se explica en cada apartado cuando considere oportuno y sin necesidad de que el profesor se encuentre presente.
- **Realizar otro tipo de actividades que no sean exclusivas de programación** donde el alumnado podrá, por ejemplo, presentar opiniones, explicar alternativas de implementación, responder cuestiones, cuestionarios etc.

## Estructuración y metodología de la actividad formativa.

**Material suministrado.** La actividad formativa consta de:

- El **documento** que actualmente estás leyendo. Sigue paso a paso el contenido. En caso de duda, contacta con tu profesor/a. **Lee todo el contenido y evita saltarte partes.**
- Un **programa de training**. Se trata de un proyecto Java en NetBeans. Este proyecto, tiene un programa principal implementado con entrada y salida estándar. Este programa, muestra un menú con diferentes opciones en las que se puede interactuar con una base de datos de ejemplo de MySQL denominada *sakila*. Las actividades formativas se basan en el código suministrado y en modificaciones solicitadas sobre él.

**Medios utilizados** durante la actividad formativa. En el documento, verás este tipo de gráficos, iconos y medios que se enumeran y explican a continuación:

	Verás este icono acompañado de un cuadro de texto en el que se introducen notas o consejos relacionados con el contenido que se está exponiendo. Presta pues especial atención a ellos.
	Cuando veas este icono, se trata de un cuestionario que deberás contestar. Se te plantearán una serie de preguntas y respuestas posibles relacionadas con el contenido que se está exponiendo. <b>Resalta en negrita</b> la respuesta o respuestas que consideres correctas.
	Cuando veas este icono, se te solicitará que ejecutes una determinada prueba para verificar el funcionamiento de una parte de la aplicación de training.

## Requisitos para ejecutar la aplicación de training.

Para poder ejecutar con éxito la aplicación suministrada en la formación, debes seguir los siguientes pasos:

- Tener **instalado y en ejecución** una instancia de servidor de base de datos **MySQL**. Si no dispones de servidor de base de datos, procede de la siguiente manera en función de tu sistema operativo:
  - o Si tienes **Linux**, **descarga desde Aules** “*Bitnami para instalar el manejador de servidores*” y, una vez descargado, sigue el PDF “*Arrancar servidores*” para arrancarlo.
  - o Si tienes **Windows**, **descárgate directamente el instalable** desde <https://dev.mysql.com/downloads/installer/>. La descarga es gratuita y no requiere registro.

- Tener **instalada la base de datos sakila** en la instancia del servidor de MySQL.
  - o Para ello, **descarga el ZIP** o TGZ de la base de datos desde <https://dev.mysql.com/doc/index-other.html>

#### Example Databases

Title	DB Download	HTML Setup Guide	PDF Setup Guide
employee data (large dataset, includes data and test/verification suite)	<a href="#">GitHub</a>	<a href="#">View</a>	<a href="#">US Ltr   A4</a>
world database	<a href="#">TGZ   Zip</a>	<a href="#">View</a>	<a href="#">US Ltr   A4</a>
<b>sakila database</b>	<b><a href="#">TGZ   Zip</a></b>	<a href="#">View</a>	<a href="#">US Ltr   A4</a>
airportdb database (large dataset, intended for MySQL on OCI and HeatWave)	<a href="#">TGZ   Zip</a>	<a href="#">View</a>	<a href="#">US Ltr   A4</a>
menagerie database	<a href="#">TGZ   Zip</a>		

Ilustración 1 - Captura de la web de MySQL con la descarga de la base de datos de ejemplo (sakila)

- o En él, encontrarás tres archivos:

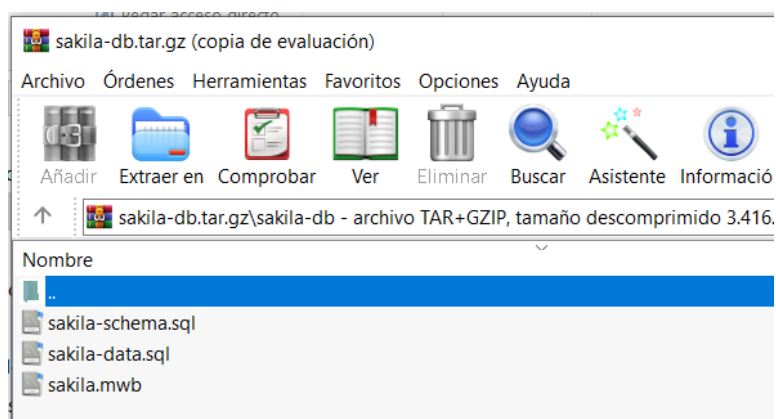
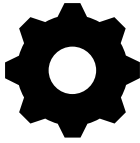


Ilustración 2 - Contenido del paquete de descarga de la base de datos de ejemplo

- Ejecuta **“sakila-schema.sql”** en el *Workbench* de MySQL para crear el esquema de la base de datos que utilizaremos en la aplicación.
- Ejecuta **“sakila-data.sql”** en el *Workbench* de MySQL para insertar los datos de prueba de las tablas utilizados en la aplicación.



Verifica que las tablas tienen datos, por ejemplo, ejecutando esta consulta desde el *Workbench* de MySQL:

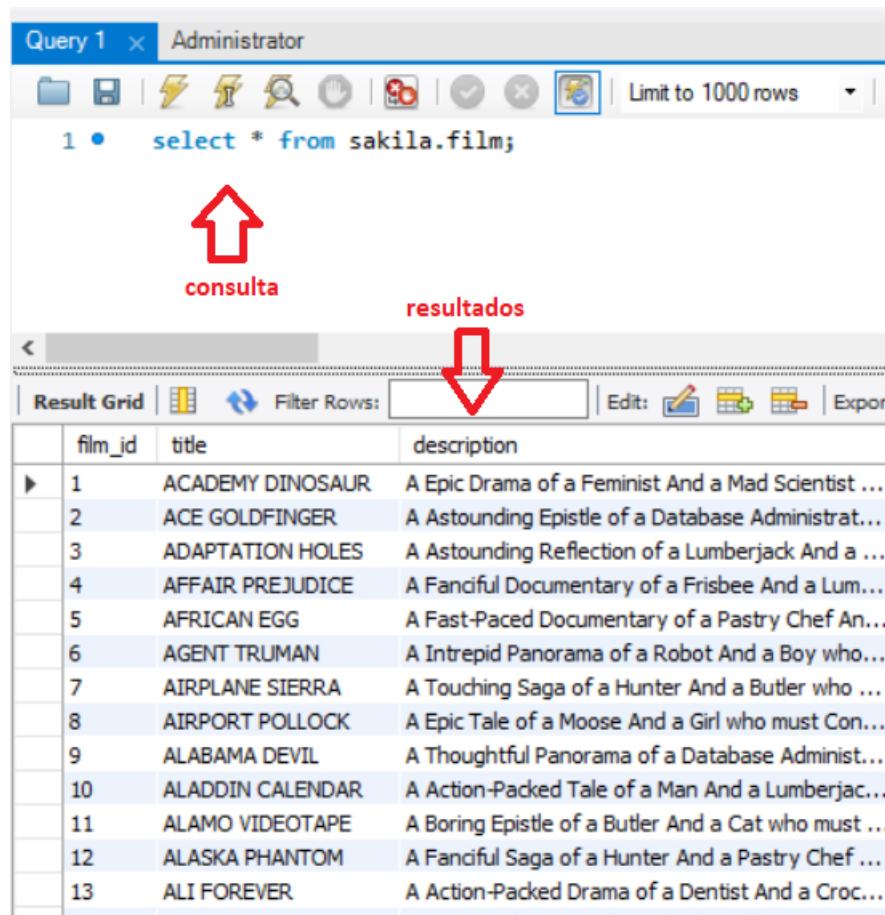


Ilustración 3 - Comprobación de la correcta instalación de la base de datos de ejemplo

- **Abre el modelo "sakila-mwb"** desde el *Workbench* de MySQL y echa un vistazo al esquema, con sus tablas, relaciones y contenidos.

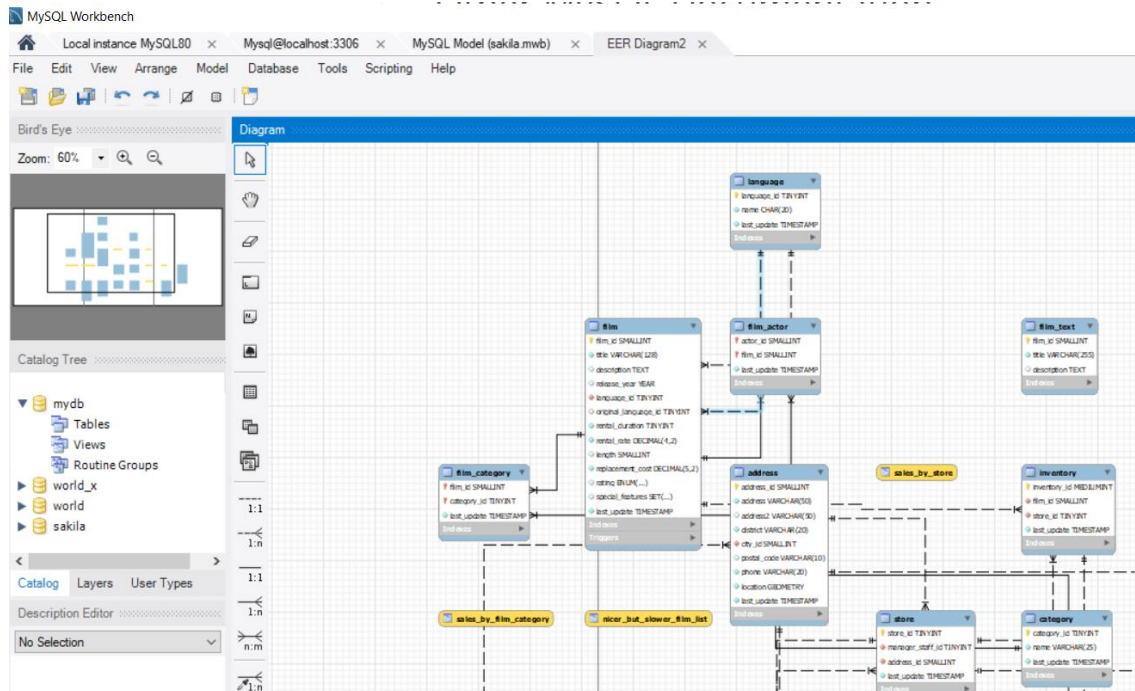


Ilustración 4 - Modelo de la base de datos de ejemplo abierto con el Workbench de MySQL



### CONSEJO

No es requerido que conozcas el esquema en profundidad, pero sí al menos la estructura de los elementos que se utilizan en el training. Las tablas implicadas son *language*, *city* y *film*. Házte un favor, y dedícale el tiempo necesario para entender esta parte del esquema.

- Tener en el *classpath* de la aplicación la **librería JDBC con el driver de MySQL**. En caso de utilizar NetBeans, la aplicación de training que has descargado ya la lleva incluida. No requieres acción alguna al respecto. En caso contrario, descárgala de abajo e inclúyela en tu IDE (Eclipse, IntelliJ, Visual Studio Code...) según corresponda.



mysql-connector-j-8.  
4.0.zip

La librería, debes verla desde el IDE. Es esta:

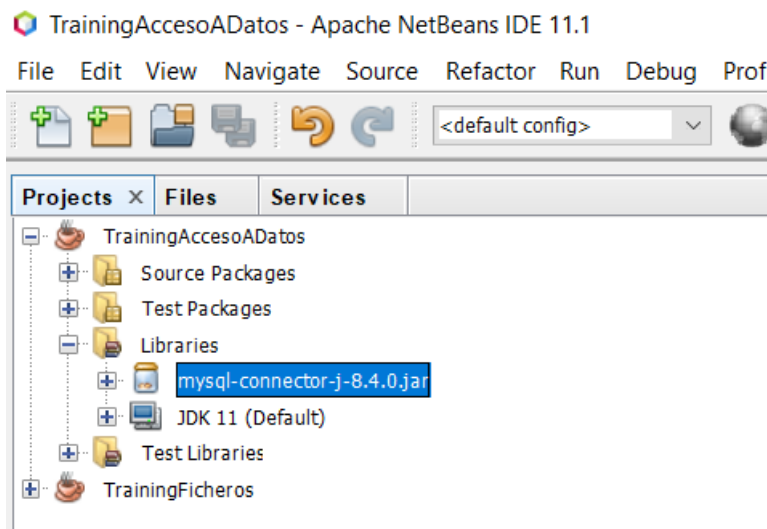
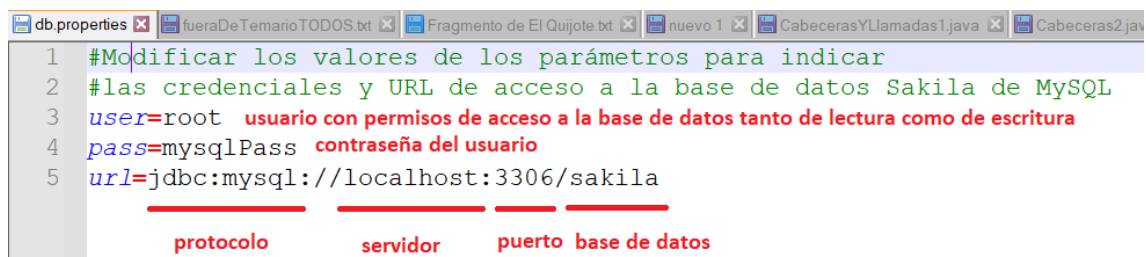
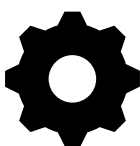


Ilustración 5 - Librería de MySQL para Java incluida en la aplicación de training

- **Configurar el acceso a datos** desde la aplicación, editando el fichero ubicado en **resources\db.properties**. Para ello, abre el archivo incluido en la aplicación de training con un editor de textos y configura los tres parámetros de conexión siguiendo estas pautas



- Si el usuario accede sin contraseña, directamente borra el contenido detrás de **pass=**.
- **jdbc:mysql** es el protocolo específico para conectar por Java a una base de datos de MySQL. No lo cambies. Otros protocolos existen para diferentes SGBD.
- El puerto utilizado es el puerto por defecto que utiliza el servidor de MySQL. Si lo tienes configurado diferente en tu servidor, edita consecuentemente el puerto en el fichero **db.properties**.



Verifica que la aplicación de training te funciona. Desde tu IDE, ejecuta el programa principal de *training.TrainingApp* y verifica que aparece el menú de opciones:



```

Outp... x Javadoc Java Call Hierarchy Search Results
Debugger Console x TrainingAccesoADatos (run-single) x TrainingAccesoADatos (run-si
Compiling 1 source file to C:\Users\jsanm\OneDrive - C
compile-single:
run-single:

Elija una opción:
    1)Consultar todas las ciudades
    2)Consultar las ciudades filtradas por LIKE
    3)Consultar películas por rango de duración
    4)Modificar descripción de una película
    5)Salir
Opción:
  
```

Ilustración 6 - Verificación de la correcta instalación de la aplicación de training

Si has llegado hasta aquí, enhorabuena, ya puedes pasar al siguiente apartado. En caso de error, habla con tu profesor/a o revisa punto por punto los pasos previos para ver si has seguido las pautas al pie de la letra.

*ACTIVIDAD 1 – CUESTIONARIO SOBRE LA TABLA CITY*

Revisa el esquema de la base de datos *sakila* y responde al siguiente cuestionario. **Resalta en negrita** la respuesta que consideres correcta. Solamente hay una válida.

1. La tabla **city**, ¿cuántos campos tiene?

Uno	Dos	Tres	Cuatro
-----	-----	------	--------

2. ¿Cuál es su **clave primaria**?

<i>city_id</i>	<i>city</i>	<i>country_id</i>	<i>last_update</i>
----------------	-------------	-------------------	--------------------

3. ¿Tiene alguna **clave ajena**?

No	Sí, <i>city_id</i> a la tabla <i>country</i>	Sí, <i>country_id</i> a la tabla <i>country</i>	Sí, <i>country_id</i> a la tabla <i>city</i>
----	--	---	--

4. ¿Algún campo **admite nulos**?

No	Sí, todos	Sí, todos excepto la clave primaria	Sí, <i>last_update</i> , por ser menos importante
----	-----------	-------------------------------------	---

5. El campo *city\_id* es de tipo **SMALLINT**, ¿cuántos bytes ocupa en MySQL? (consulta si es necesario por internet)

Uno	Dos	Tres	Cuatro
-----	-----	------	--------

6. En base a la respuesta anterior, ¿qué tipo de datos Java sería más idóneo para almacenar un valor del campo *city\_id* en una variable Java?

short	int	long	String
-------	-----	------	--------

7. ¿Qué tipos Java utilizarías para almacenar los valores de cada campo?

<i>city_id</i> : int <i>city</i> : String <i>country_id</i> : int <i>last_update</i> : String	<i>city_id</i> : short <i>city</i> : String <i>country_id</i> : String <i>last_update</i> : Date	<i>city_id</i> : short <i>city</i> : String <i>country_id</i> : short <i>last_update</i> : Date
--	---	--



#### PAUTAS DE DISEÑO

Conocer bien los detalles de diseño de la base de datos es indispensable a la hora de implementar una aplicación que manipule dichos datos de forma correcta. Del conocimiento adquirido con el estudio de la estructura de los datos podemos:

- Crear clases java para las diferentes entidades / tablas. Ej: tendremos una clase *City* que se corresponde con la tabla *city*.
- Los objetos o instancias de la clase Java serán equivalentes a los registros de la tabla. Ej: un objeto *City* concreto contendrá la información de un registro de ciudad de la tabla *city*.
- Cada clase (entidad) tendrá como propiedades los campos de la tabla a la que representan. Ej: la clase *City* tendrá las propiedades *cityId*, *city*, *lastUpdate*, equivalentes a *city\_id*, *city* y *last\_update*, respectivamente.
- Los tipos de las propiedades dependerán del tipo de datos Java más apropiado para almacenar los valores del campo al que representan. Así pues, un *VARCHAR* de MySQL precisa un tipo de datos *String* en Java, por ejemplo, para manejar los valores posibles.

	<ul style="list-style-type: none"> <li>- Las restricciones de valores no nulos se pueden implementar en Java validando el parámetro del método set&lt;Propiedad&gt;( ) correspondiente.</li> <li>- Las relaciones 1 a 1 se representan en Java como "clases que contienen clases".</li> <li>- Las relaciones 1 a muchos, se representan en Java con el patrón <i>composite</i> ya estudiado en la unidad de colecciones dinámicas.</li> <li>- Hay muchas otras buenas prácticas...</li> </ul>
--	---

Cierto es que hay mucha información sintetizada en el cuadro anterior. Iremos procesando y consolidando poco a poco la adquisición de estas buenas pautas de diseño.

```
/**
 * Clase para representar cada registro de la tabla <code>city</code> en la aplicación
 * de training
 * @author jsanm
 */
public class City {
    private Short cityId;
    private String city;
    private Short countryId; //este atributo podría ser de tipo Country (no definido)
    private Date lastUpdate;
}
```

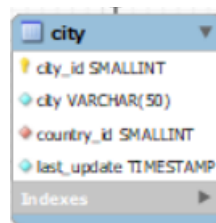


Ilustración 7 - Clase java City y tabla city equivalentes

## Información sobre la aplicación de training.

### Contenido de la aplicación.

- a) La aplicación consta de **4 paquetes** bien diferenciados:

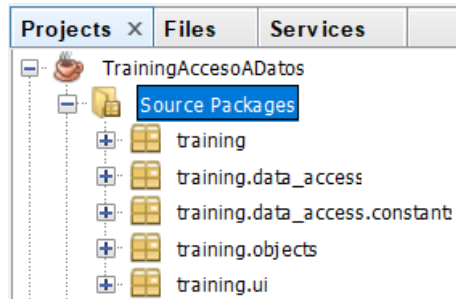
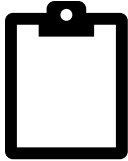


Ilustración 8 - Paquetes de la aplicación de training

- **training.** El **paquete raíz, que contiene las aplicaciones** que puedes ejecutar. En concreto, *TrainingApp.java* (entrada/salida estándar) y *TrainingUIApp.java* (con interfaz de usuario).
- **training.objects.** **Paquete con las entidades de negocio.** Contiene las clases Java o entidades equivalentes a las tablas de la base de datos. Se han implementado tres de ellas: *Language*, *Film* y *City*.
- **training.data\_access.** **Paquete de acceso a datos.** Contiene las clases Java necesarias para conectar a la base de datos, ejecutar consultas y manipular los datos.
- **training.ui.** **Paquete con la interfaz de usuario.** Contiene las clases con los formularios y el manejo de eventos. Lo dejamos para el final.



### ACTIVIDAD II – CUESTIONARIO SOBRE LA PAQUETIZACIÓN

Revisa por encima el contenido de cada paquete Java y contesta el siguiente cuestionario. **Resalta en negrita** la respuesta que consideres correcta. Solamente hay una válida.

1. En el paquete de **entidades de negocio**, ¿logras encontrar **sentencias SQL**?

Sí	No
----	----

2. En el paquete de entidades de negocio, ¿**hay referencias a tablas** o nombres de columna (aparte de en los comentarios y en el JAVADOC)?

Sí	No
----	----

3. En el paquete de **acceso a datos**, ¿logras encontrar **sentencias SQL**?


Sí	No
----	----

4. En el paquete de acceso a datos, ¿**hay referencias a tablas** o nombres de columna?

Sí	No
----	----

5. En el paquete de acceso a datos, ¿**encuentras implementada la conexión** y cierre de la misma con la base de datos (en concreto en la clase *DataAccessManager*)?

Sí	No
----	----

	<p style="text-align: center;">Patrón de diseño DAO</p> <p>El patrón de diseño DAO o de Objeto de Acceso a Datos (de sus siglas en inglés) es muy útil en el desarrollo de aplicaciones que manipulan datos persistidos en una base de datos y su uso está altamente extendido. Se recomienda, pues, seguir este patrón de diseño o utilizar <i>frameworks</i> que son fieles a él, como <i>Hibernate</i>, <i>Spring</i> o <i>MyBatis</i>.</p> <p>Este patrón de diseño pretende independizar una aplicación de la lógica necesaria para acceder a los datos (implementada EXCLUSIVAMENTE en el paquete <i>data_access</i> en el training).</p> <p>Algunas de las características más importantes de este patrón de diseño son:</p> <ul style="list-style-type: none"> <li>- Separación de la lógica de negocio y de la lógica de acceso a datos.</li> <li>- Capacidad de alterar el origen de datos sin variar la lógica de negocio.</li> <li>- Independencia de la aplicación de la tecnología subyacente de persistencia.</li> <li>- Arquitectura por capas.</li> <li>- Mayor complejidad de desarrollo.</li> <li>- Habitualmente, peor rendimiento.</li> </ul>
--	--

b) El proyecto de NetBeans consta de las siguientes **carpetas**:

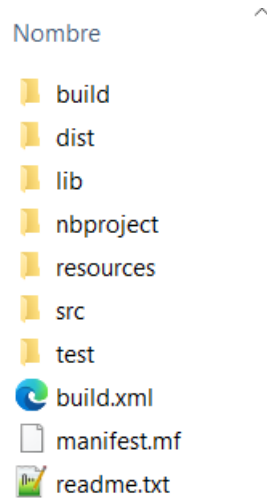


Ilustración 9 - Contenido de la aplicación

- Como en todo proyecto *Ant* de NetBeans:
  - o **src**, donde reside el código fuente de las clases Java.
  - o **test**, donde residen las clases de test (no hay en este proyecto).
  - o **build**, donde están las clases compiladas (*bytecodes*).
  - o **dist**, donde reside la librería JAR generada tras la compilación.
- Carpeta **resources**, que incluye el ya mencionado fichero *db.properties* en el que se configura el acceso a la base de datos.
- Carpeta **lib**, que incluye la ya mencionada librería JDBC de MySQL.
- Fichero **readme.txt**, con las instrucciones de instalación de la aplicación.

## Arquitectura.

La aplicación de training sigue una **arquitectura de 3 capas**. En este tipo de arquitecturas, se separa la lógica por partes:

- La **Capa de Presentación** (paquetes *training* y *training.ui*) contiene la interacción con el usuario, tanto la entrada como la salida. Manipula datos de la capa de negocio como películas, ciudades, etc y las muestra/solicita al usuario. El grueso del trabajo a realizar por tu parte en la unidad 11 sobre interfaz de usuario está aquí.
- La **Capa de Negocio** (paquete *training.objects*) dispone de las entidades de negocio así como la lógica de la aplicación. Dado que es una aplicación de training, no dispone de mucha chicha... se la tendrás que dar tú 😊
- La **Capa de Acceso a Datos** (paquete *training.data\_access*) contiene los DAO de acceso a la base de datos. Hay uno por tabla (*CityDAO*, *LanguageDAO* y *FilmDAO*), donde se publican operaciones que reciben parámetros de negocio y devuelven resultados de negocio: las traducciones y transformaciones correspondientes se realizan aquí. El grueso del trabajo a realizar por tu parte en la unidad 10 está aquí.

Nótese que es la capa de acceso a datos la que manipula, gracias a la **librería mysql-connector-j.X.X.X.jar**, el acceso a los datos. Esta librería es proporcionada por la propia compañía MySQL para acceder desde programas Java a los datos.

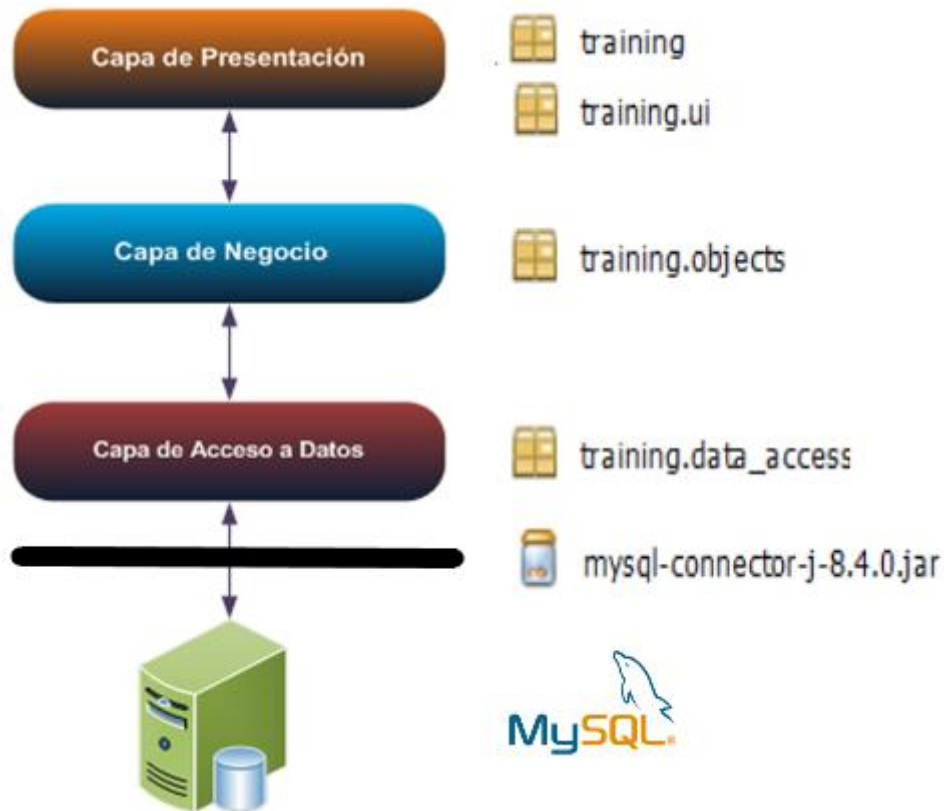


Ilustración 10 - Arquitectura por capas de la aplicación

## Parte I – Acceso a datos con Java.

### Librería jdbc de MySQL.

Como hemos indicado en el apartado de Arquitectura., la librería *mysql-connector-j-8.4.0.jar* maneja el acceso a los datos. Se trata pues, de un **API Java o driver de la base de datos**. En concreto, Oracle distribuye en ella una implementación Java de las operaciones básicas sobre MySQL para:

- **Conectar y desconectar** a una base de datos desde una aplicación.
- **Ejecutar** desde ella **sentencias SQL** y recuperar los resultados.
- **Ejecutar otras sentencias DML** (*update, delete, insert*).
- Ejecutar sentencias DDL (*create table, create index...*). Fuera de alcance de esta unidad temática.
- Ejecutar otras operaciones (vistas, procedimientos almacenados...). Fuera de alcance de esta unidad temática.

Indicar pues que esta unidad temática se va a centrar exclusivamente en los tres primeros guiones.

Añadir la librería jdbc de MySQL a un proyecto en NetBeans.

La aplicación de training ya tiene incluida en el *classpath* la librería jdbc. No obstante, si desarrollas una *app* de acceso a datos desde cero en NetBeans, deberás seguir estos pasos antes de empezar a trabajar con la base de datos.

1. Expande el nodo *Libraries* (bibliotecas) del proyecto al que se le va a agregar la librería para ver las que actualmente tiene agregadas:

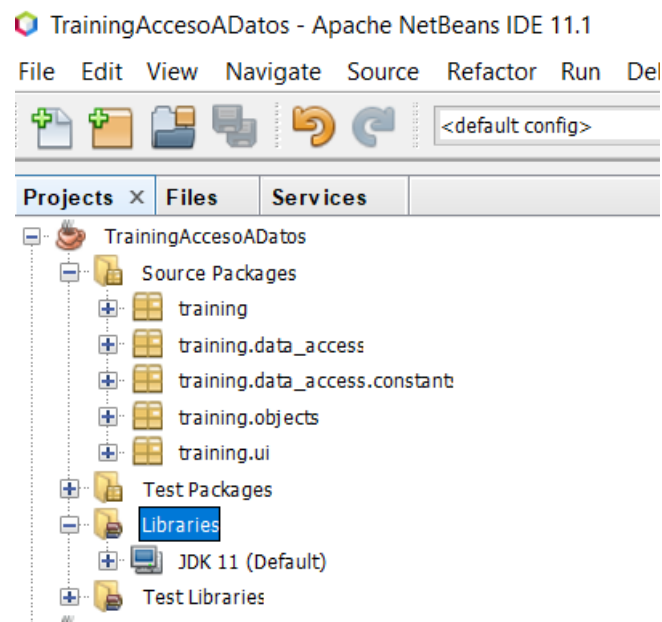


Ilustración 11 - Librerías de un proyecto NetBeans



2. Selecciona el proyecto y en su menú contextual, haz clic en la opción *Properties*:

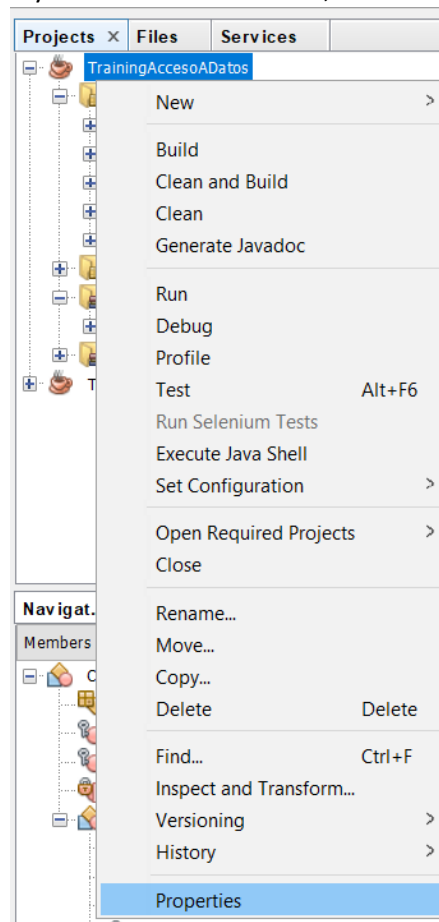


Ilustración 12 – Ver las propiedades de un proyecto en NetBeans

3. Aparece un cuadro de diálogo con las diferentes propiedades del proyecto, agrupadas por secciones. Selecciona la sección *Libraries* en el panel izquierdo:

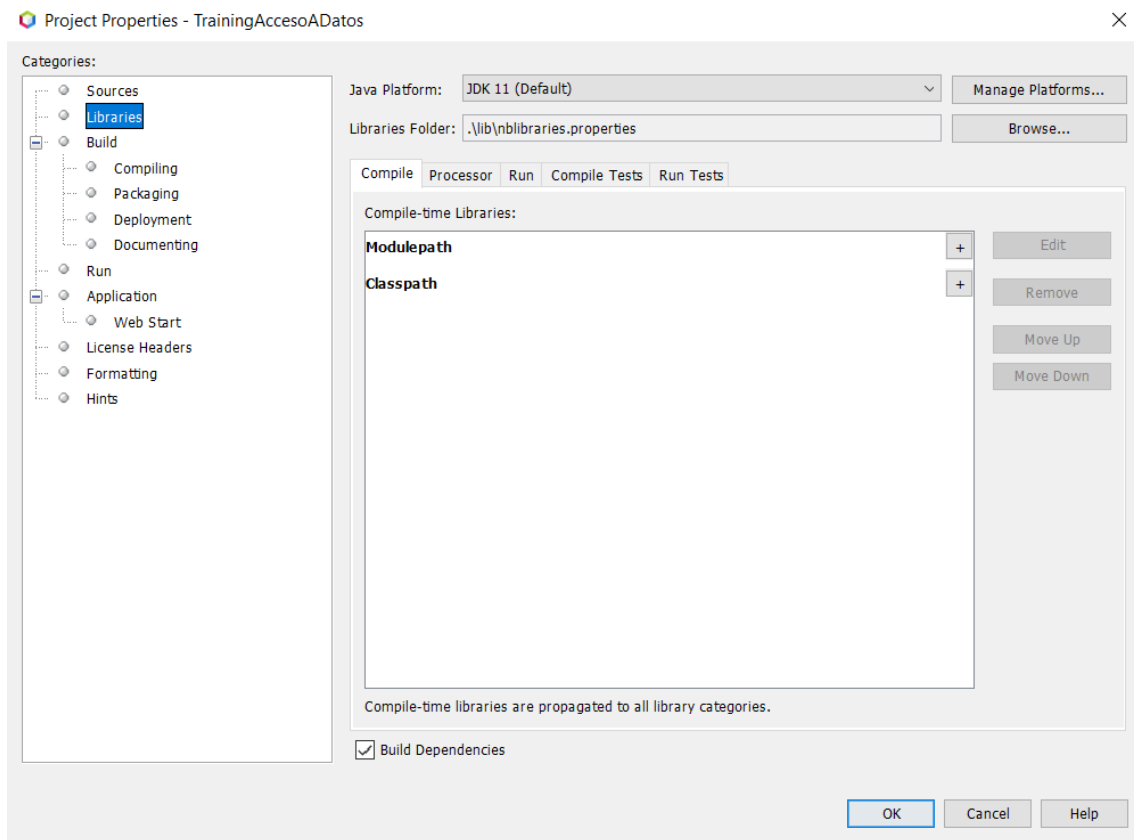


Ilustración 13 - Configurar librerías de un proyecto

4. Presiona el botón + que aparece a la derecha del epígrafe *Classpath* y, en el menú contextual que aparece, selecciona la opción *Add JAR/Folder*.

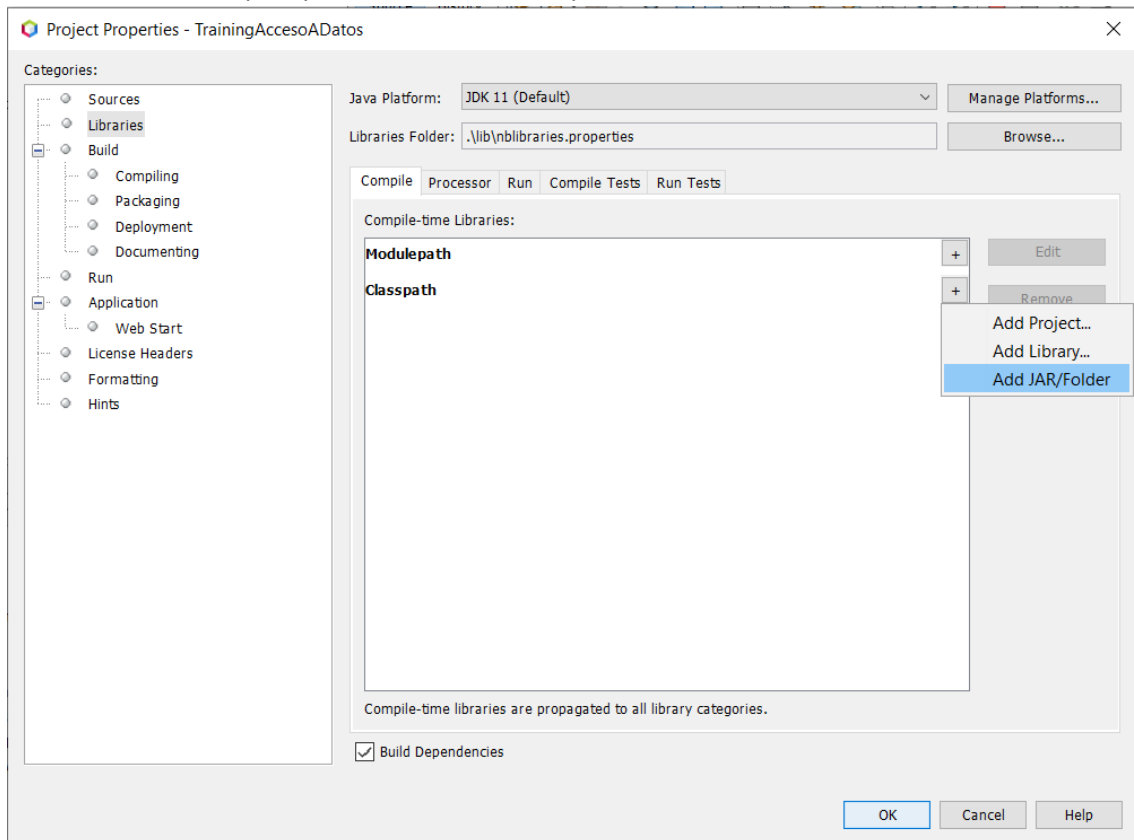



Ilustración 14 - Añadir una librería externa al classpath

	El <i>classpath</i>
	<p>El <i>classpath</i> es una variable de la JDK con las rutas a las diferentes librerías externas y carpetas contenedoras de clases a las que hace referencia una aplicación Java. Puedes verlo como símil de la variable PATH que utilizan los intérpretes de comandos de los sistemas operativos para encontrar los comandos que puede ejecutar.</p> <p>Así pues, para que la aplicación funcione con normalidad, en tiempo de ejecución las librerías configuradas en el <i>classpath</i> deben estar accesibles.</p>

- Se abrirá un cuadro de diálogo para que selecciones la librería a añadir. Navega hasta la carpeta *lib* del proyecto de training y selecciona la librería *mysql-connector-j-8.4.0.jar*. Mantén la referencia relativa al directorio. Finalmente, pulsa el botón *Open*.

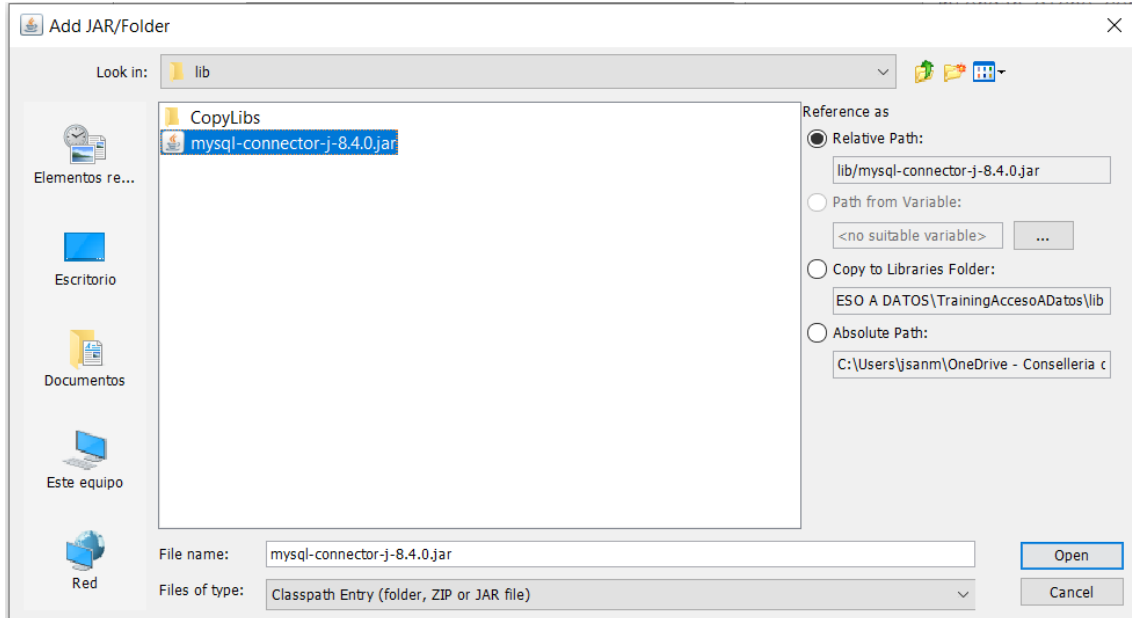


Ilustración 15 - Selección del driver jdbc de MySQL

- Si todo ha ido bien, en el explorador de proyectos debería aparecer la librería añadida al *classpath*.

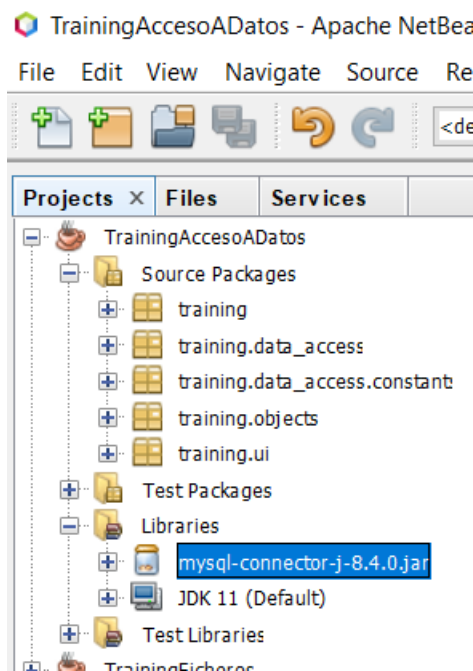


Ilustración 16 - Librería añadida al classpath

## Interfaces Java necesarias para conectar por jdbc a una base de datos.

Todo driver Java de acceso a datos por jdbc debe implementar varias interfaces. Las más importantes son:

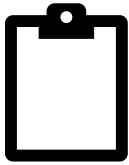
- *Connection*. Esta clase permite conectar con una base de datos, gestionar la transaccionalidad y las credenciales de acceso.
- *PreparedStatement*. Esta clase permite definir una sentencia SQL (*select*, *insert*, *update* y *delete*) como un String y ejecutarla sobre una conexión a base de datos.
- *ResultSet*. Esta clase nos permite obtener los resultados de una sentencia SQL y manipularlos en la aplicación Java.



### CONSEJO

Echa un vistazo al JAVADOC de estas tres interfaces. Te resultará de gran utilidad.

## Conexión a base de datos. Interfaz *Connection*.



### ACTIVIDAD III – CONEXIÓN A UNA BASE DE DATOS

Revisa la clase *DataAccessManager* de la aplicación. En concreto, revisa la función *createConnection()*. Revisa también los llamantes de la función. Puedes identificarlos utilizando la opción “*call hierarchy*” del menú contextual. Tras ello, contesta el siguiente cuestionario. **Resalta en negrita** la respuesta que consideres correcta. Solamente hay una válida.

1. El objeto `java.sql.Connection`, ¿se crea siempre al crear un objeto *DataAccessManager*?

Sí	No
----	----

2. ¿Cuántos objetos *DataAccessManager* se crean en la aplicación de training?

0	1	2	3
---	---	---	---

3. Por ende, ¿cuántas conexiones a base de datos se crean en la aplicación de training?

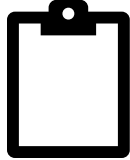
0	1	2	3
---	---	---	---

4. Si tuvieras que elegir la más adecuada, ¿en qué momento se crea una conexión a la base de datos de training?

Antes de ejecutar cada sentencia SQL	Al iniciarse la aplicación	Antes de persistir los cambios de la capa de negocio	Cada vez que una conexión se cierra, se abre una nueva
--------------------------------------	----------------------------	--	--

5. ¿Es necesario conocer cómo implementa el driver la interfaz Connection?

SI	NO
----	----



#### ACTIVIDAD IV – DESCONEXIÓN DE UNA BASE DE DATOS

Revisa la clase *DataAccessManager* de la aplicación. En concreto, revisa la función *close()*. Revisa también los llamantes de la función. Si intentas identificarlos utilizando la opción "*call hierarchy*", ¡verás que no hay ninguno! Sin embargo, el método *close()* pertenece a la interfaz *AutoCloseable*. Consúltala en internet e intenta averiguar su relación con la instrucción *try-con-recurso*. Tras ello, vuelve a inspeccionar el código fuente y contesta el siguiente cuestionario. **Resalta en negrita** la respuesta que consideres correcta. Solamente hay una válida.

1. El método ***java.sql.Connection.close()***, ¿se invoca explícitamente desde la aplicación de training?

Sí	No
----	----

2. ¿Cuántas veces se ejecuta durante la ejecución del aplicativo?


0	1	2	Indefinidas
---	---	---	-------------

3. Si tuvieras que elegir la más adecuada, ¿en qué momento se cierra una conexión a la base de datos de training?

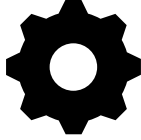
Tras ejecutar cada sentencia SQL	Antes de finalizarse la aplicación	Justo después de persistir los cambios de la capa de negocio
----------------------------------	------------------------------------	--

Tras haber respondido a los cuestionarios relacionados con las aperturas y cierres de conexiones a base de datos, debes haberte percatado de los siguientes matices. Los resalto, para que no pasen desapercibidos ya que son de gran importancia:

- Los métodos de la clase java ***DataAccessManager*** ***createConnection()*** y ***close()*** no son estándar, **son propios del aplicativo**. Tenlo en cuenta, no los intentes invocar en otros aplicativos porque no estarán. Que no te líe que estén en inglés; los he definido así a conciencia.
- **Los realmente estándar** que necesitarán toda aplicación Java de acceso a datos **son *DriverManager.createConnection()* y *Connection.close()*** que están envueltos por los dos anteriores.
- El motivo de *wrappear* la creación y liberación de las conexiones es para que **toda la aplicación utilice una única conexión**, creada al arrancar y liberada justo antes de terminar, de forma segura. Esto se consigue, a su vez, aplicando otras técnicas y patrones de diseño:
  - **Singleton**. Sólo se crea un único objeto *DataAccessManager*. Por ello su constructor es privado y el método *createConnection()*, también y así controlar sus invocaciones; sólo una.
  - **Una conexión a base de datos es un recurso importante** en toda aplicación, ya que mantiene abierto en memoria un flujo de datos de entrada y salida hacia un elemento externo a la app. Similar a cómo ocurría con los flujos de lectura y escritura de ficheros. Así pues, toda conexión **se debe manejar de forma segura**, con instrucciones *try/catch/finally* que aseguren que nunca se queda una conexión abierta aunque la aplicación falle.
  - **Abrir una nueva conexión es una operación costosa** temporalmente (hay que habilitar un canal de comunicación, validar credenciales...). Queda a tu elección en tus aplicaciones abrir una conexión antes de ejecutar cada sentencia SQL, abrir una única al inicio como en la aplicación de training, o temporalmente cerrarlas por precaución y abrir nuevas.

	<p>NOTA</p> <p>Indicar que la instrucción <i>try-con-recurso</i> también la puedes utilizar con los recursos <i>BufferedWriter/BufferedReader</i> para manejar un flujo de ficheros de forma segura y que se auto cierre al finalizar el bloque <i>try</i>, sin necesidad de implementar el bloque <i>finally</i> (es implícito).</p>
---	---

Ejecución de consultas. Interfaces *PreparedStatement* y *ResultSet*.

	<p>Ejecuta la aplicación <i>TrainingApp</i> y elige la opción de menú #1 "Consultar todas las ciudades". Debería aparecer el nombre de todas (son 600) las ciudades de la base de datos.</p>
---	--

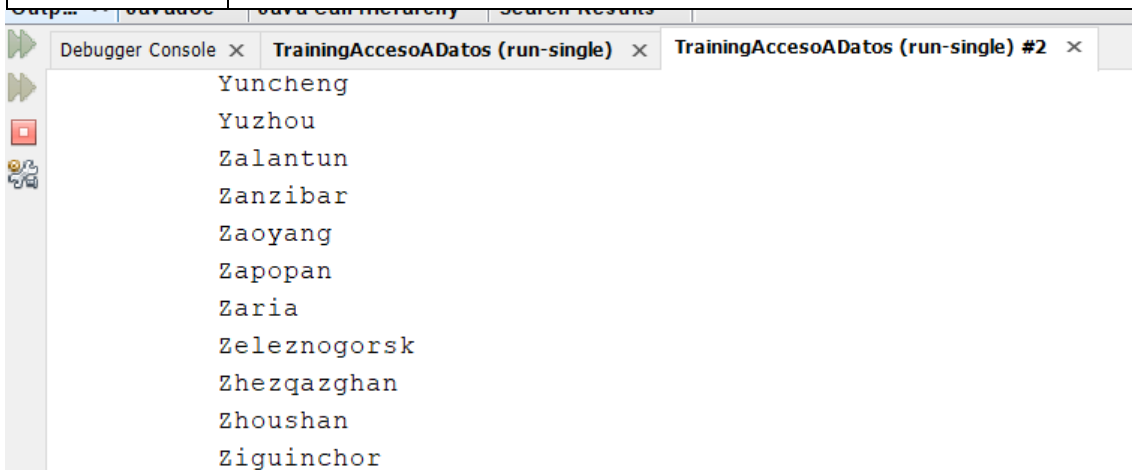

  


Ilustración 17 - Últimos registros mostrados de la consulta de ciudades

	<p style="text-align: center;"><i>ACTIVIDAD V – CONSULTA SIN PARÁMETROS</i></p> <p>Revisa la función que se ejecuta en la consulta de ciudades. Está en <i>CityDAO.loadAllCities()</i> (.). Luego, contesta el siguiente cuestionario. <b>Resalta en negrita</b> la respuesta que consideres correcta. Solamente hay una válida salvo que se indique lo contrario.</p>
---	--

1. Revisando la cabecera de la función, ésta **devuelve una lista de...**

Objetos del paquete java.sql	Objetos de la librería de MySQL	Objetos de la capa de negocio
---------------------------------	------------------------------------	----------------------------------



2. El encargado de realizar la recuperación de la información de la base de datos, es la operación *PreparedStatement.executeQuery()*

SI, devolviendo un ResultSet, que es un cursor con un puntero a una fila de una selección de registros	NO
--	----

3. ¿A qué **patrón de diseño** se asemeja la iteración sobre el objeto *ResultSet* para generar los objetos *City*?

<i>Composite</i>	A ninguno	Iterator	<i>Wrapper</i>
------------------	-----------	----------	----------------

4. Según la función estática *readCityFromResultSet()*, parece necesario conocer el nombre de las columnas devueltas por un ResultSet, pero... ¿**Cómo puedes acceder a la columna concreta de la fila apuntada por un cursor**? Varias respuestas son válidas.

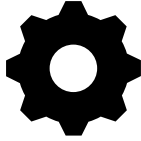
Por nombre de columna o alias	Por índice de columna	No es posible acceder
-------------------------------	-----------------------	-----------------------

5. Prueba a cambiar la sentencia SQL por "SELECT **S** \* FROM city". ¿**Qué ocurre al cometer un error de sintaxis en la sentencia SQL**?

No compila	Compila y da los mismos resultados que antes	Falla la operación <i>prepareStatement</i>	Falla la operación <i>executeQuery</i>
------------	--	--	--

6. Prueba a cambiar la sentencia SQL por "SELECT \* FROM city~~yyy~~". ¿**Qué ocurre al cometer un error semántico en la sentencia SQL**?

No compila	Compila y da los mismos resultados que antes	Falla la operación <i>prepareStatement</i>	Falla la operación <i>executeQuery</i>
------------	--	--	--

	<p>Ejecuta la aplicación <i>TrainingApp</i> y elige la opción de menú #3 "Consultar películas por rango de duración". El sistema te solicitará un rango de minutos y mostrará las películas cuya duración esté comprendida en ese rango de tiempo.</p>
---	--


Opción: 3

Escriba la duración desde: 80

Escriba la duración hasta: 81

```
CLASH FREDDY - 81' - English
DESPERATE TRAINSPOTTING - 81' - English
FLAMINGOS CONNECTICUT - 80' - English
GARDEN ISLAND - 80' - English
LOSER HUSTLER - 80' - English
MEET CHOCOLATE - 80' - English
PEAK FOREVER - 80' - English
ROMAN PUNK - 81' - English
SADDLE ANTITRUST - 80' - English
SEA VIRGIN - 80' - English
SHAWSHANK BUBBLE - 80' - English
TURN STAR - 80' - English
WAR NOTTING - 80' - English
```

*Ilustración 18 – Películas filtradas por rango de duración*

	<p style="text-align: center;"><i>ACTIVIDAD VI – CONSULTA CON PARÁMETROS</i></p> <p>Revisa la función que se ejecuta en la consulta de películas por rango de duración. Está en <i>FilmDAO.loadFilmsByLengthRange()</i>. Luego, contesta el siguiente cuestionario. <b>Resalta en negrita</b> la respuesta que consideres correcta. Solamente hay una válida.</p>
---	---

1. Revisando la cabecera de la función, éste **recibe como parámetros** ...


Objetos del paquete java.sql	Objetos de la librería de MySQL	Parámetros relacionados con la capa de negocio
---------------------------------	------------------------------------	--

2. ¿Por qué crees que se utilizan los interrogantes en la generación del *PreparedStatement*?

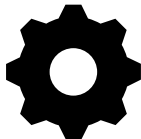
No se deben utilizar, se trata de un error de sintaxis	Cada interrogante es un parámetro de la consulta y con ellos, los parámetros de la función <i>loadFilmsByLengthRange</i> se mapean automáticamente	Cada interrogante es un parámetro de la consulta, pero antes de ejecutarla, se debe mapear cada interrogante con su valor correspondiente con el método <i>setTipo</i>
--	--	--

3. Según la función estática *readFilmFromResultSet()*, ¿por qué crees que se utilizan tipos de datos objeto-valorados como *Short*, *Integer*, *Float* y *Double*, en lugar de los tipos básicos *short*, *int*, *float* y *double*?

Para fastidiarte y ponértelo aún más difícil todavía	Para poder representar un valor nulo de la correspondiente columna de la base de datos
--	--

	<p>CARGA EN CASCADA</p> <p>No sé si te has percatado, pero la relación N a 1 entre un registro película (film) e idioma (language), se ha implementado en Java cargando en cascada para cada objeto película (Film) su correspondiente idioma (Language). Para ello, un DAO puede hacer uso de la funcionalidad expuesta por otro DAO, como es el caso.</p>
---	---

### Actualización de registros.

	Ejecuta la aplicación <i>TrainingApp</i> y elige la opción de menú #4 "Modificar descripción de una película". El sistema te solicitará el título de una película y, si existe, te mostrará su información relevante junto a la descripción actual. Acto seguido, te pedirá su nueva descripción y la cambiará en base de datos.
---	--

Opción: 4

Escriba el título de la película: WAR NOTTING

WAR NOTTING - 80' - English

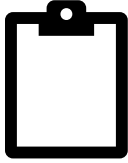
Descripción actual:

A Boring Drama of a Teacher And a Sumo Wrestler who must Challenge a Secret Agent

Escriba la nueva descripción: Nueva descripción!!!!

Registro actualizado con éxito

Ilustración 19 - Ejecución del cambio de descripción de una película



### ACTIVIDAD VII – ACTUALIZACIÓN DE REGISTROS

Revisa la función que se ejecuta en la actualización de la descripción de una película. Está en *FilmDAO.updateFilm()*. Luego, contesta el siguiente cuestionario. **Resalta en negrita** la respuesta que consideres correcta. Solamente hay una válida salvo que se indique lo contrario.

1. Revisando la cabecera de la función, éste **recibe como parámetros...**

Objetos del paquete java.sql	Objetos de la librería de MySQL	Objetos de la capa de negocio
---------------------------------	------------------------------------	----------------------------------

2. El **encargado de realizar la actualización** de la información en base de datos, es la operación ***PreparedStatement.executeUpdate()***

SI, devolviendo el número de registros afectados	NO
--	----

3. ¿**Por qué crees que se lanza una excepción** si no se actualiza un único registro? Varias opciones son válidas

Es redundante, no hace falta	Preventivamente, por si la sentencia SQL no actualiza ningún registro	Preventivamente, por si la sentencia SQL actualiza más de un registro y evitar así que persistan los cambios (haciendo <i>rollback</i> )
---------------------------------	--	---

### Eliminación e inserción de registros.

Muy fáciles; son iguales que las actualizaciones (con instrucciones SQL DELETE y INSERT, respectivamente), a través de la función *PreparedStatement.executeUpdate()*. ¡Pruébalas!

## Parte II – Interfaz de usuario con Java.

Pendiente