



## **Configuración del archivo pom. JDBC acceso a bases de datos.**

### **Contenidos**

<b>Introducción</b>	<b>2</b>
<b>Java Database Connectivity (JDBC)</b>	<b>2</b>
<b>XML</b>	<b>3</b>
Estructura de un documento XML	3
<b>Maven</b>	<b>5</b>
Creación de un proyecto Maven	7
Configuración del archivo pom.xml	10
¿Qué versión de MySQL tengo instalada?	13
Ejemplo de conexión a una base de datos	16

## Introducción

Una vez creada la Base de Datos del sistema debemos acceder a ella desde algún lenguaje de programación, en este curso estamos utilizando Java.

Java Database Connectivity (Conectividad a bases de datos de Java), más conocida por sus siglas JDBC, es *una API (Application Programming Interface) que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java*, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Para la primera conexión a Base de Datos lo que suele hacerse es descargar el controlador de la base de datos y a continuación ponerlo en el proyecto (en algunas ocasiones también lo puede proveer el IDE). Lo más correcto es usar un *gestor de dependencias*, para este curso utilizaremos: *Maven*.

Antes de empezar a ver conceptos sobre Maven, es necesario conceptualizar sobre JDBC y el lenguaje de marcas XML.

## Java Database Connectivity

El API JDBC se presenta como una colección de *interfaces Java* y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión; para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de



tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

## **XML**

XML, por sus siglas en inglés de eXtensible Markup Language, traducido como 'Lenguaje de Marcado Extensible' o 'Lenguaje de Marcas Extensible', es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

En este curso nuestro interés en XML se centra específicamente en un archivo: pom.xml.

### **Estructura de un documento XML**

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes



se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de este como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma **<nombre>**, donde *nombre* es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo para entender la estructura de un documento XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
  <Mensaje>
    <Remitente>
      <Nombre>Nombre del remitente</Nombre>
      <Mail>Correo del remitente </Mail>
    </Remitente>
    <Destinatario>
      <Nombre>Nombre del destinatario</Nombre>
      <Mail>Correo del destinatario</Mail>
    </Destinatario>
    <Texto>
      <Asunto>
        Este es mi documento con una estructura muy sencilla
        no contiene atributos ni entidades...
      </Asunto>
      <Parrafo>
        Este es mi documento con una estructura muy sencilla
        no contiene atributos ni entidades...
      </Parrafo>
    </Texto>
  </Mensaje>
</Edit_Mensaje>
```

Imagen 1. Ejemplo de un archivo XML sencillo

El código muestra en verde las *etiquetas* o *tags*. Cada etiqueta tiene un contenido con información y a su vez puede haber etiquetas dentro de otras etiquetas con sus respectivas etiquetas de cierre. En el siguiente código se puede ver más claramente lo que es una etiqueta de apertura (<Parrafo>) y su correspondiente etiqueta de cierre (</Parrafo>).

```
<Parrafo>
  Este es mi documento con una estructura muy sencilla
  no contiene atributos ni entidades...
</Parrafo>
```

Imagen 2. Ejemplo de etiqueta de apertura y cierre.

Además, la etiqueta *parrafo* (*sin acento*) tiene un contenido. Veremos con más detalle este tipo de documentos cuando veamos el documento de configuración *pom.xml*.

## Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation.

Cuando desarrollamos un proyecto sin Maven, nos empiezan a surgir muchas preguntas. Tales como:

- ¿Cómo me instalo el proyecto?
- ¿Por qué cada proyecto tiene una estructura distinta?
- ¿Cómo modificamos alguna versión de una librería?
- ¿Cómo metemos una nueva librería?
- ¿Cómo lo probamos?

Las respuestas a estas preguntas eran complejas y diferentes para cada proyecto.

La finalidad principal con la que Jason van Zyl desarrolló Maven para la empresa Sonatype (año 2002), fue aglutinar todas esas preguntas creando una manera de gestionar y construir proyecto que otorgara un conjunto de convenciones para que las preguntas anteriores y algunas más tuvieran una misma respuesta. Y también otorgar simplicidad a la creación y gestión de proyectos. Maven estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la *Apache Software Foundation*.

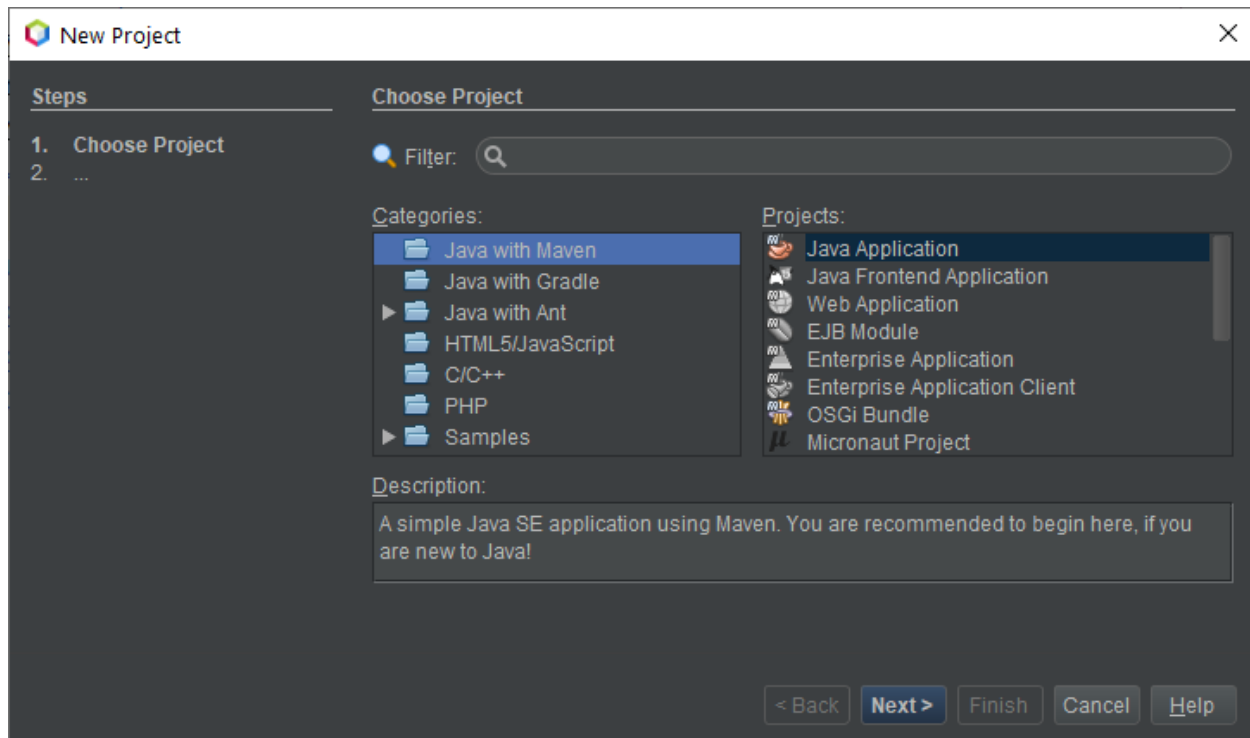
Al contrario que otras herramientas anteriores y más limitadas como *Apache Ant* (también muy popular), Maven utiliza convenciones sobre dónde colocar ciertos archivos para el proceso de construcción de un proyecto. Además, es una herramienta declarativa. Es decir, todo lo que definamos (dependencias en módulos y componentes externos, procesos, orden de compilación, plugins del propio Maven) se almacena en un archivo XML que Maven lee a la hora de funcionar. Otras alternativas, como Gradle no utilizan archivos XML, sino de otro tipo, pero usan los mismos conceptos de Maven. Maven utiliza un Project Object Model (POM.xml) para dentro de él especificar las diferentes **dependencias** o **librerías** que serán necesarias incluir en el proyecto que se esté desarrollando. Algunos ejemplos de IDEs que ya incluyen Maven por defecto en la creación de proyectos Java son: IntelliJ IDEA y Apache NetBeans.

Una característica clave de Maven es que está listo para usar en red (internet). El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

## Creación de un proyecto Maven

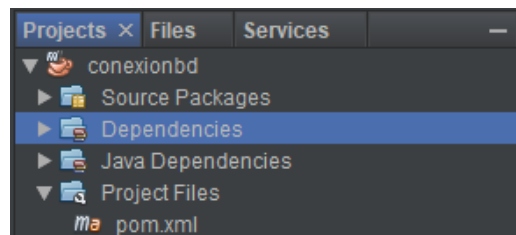
Maven puede instalarse en el sistema operativo por aparte y accederse mediante una consola de comandos (cmd de windows o bien una terminal linux). Sin embargo, en este curso solo utilizaremos el paquete que viene embebido en el IDE Apache Netbeans. La creación de un proyecto Maven en *Apache Netbeans* no dista demasiado de lo que es un proyecto común de Java, lo hacemos dirigiéndonos al menú *file* y luego

*new project*, esto muestra la ventana de la siguiente figura en donde dentro de *Java with Maven* (en lugar de *Java with Ant* por ejemplo) elegimos *Java Application*.



*Imagen 3. Creación de proyectos Maven con Apache Netbeans*

Podemos notar que este nuevo proyecto tiene ahora una carpeta llamada *Dependencies*, como se muestra en la siguiente figura.



*Imagen 4. Carpeta Dependencies de Maven*

Al final de la imagen, también se debe notar la presencia del archivo de configuración de dependencias: *pom.xml*. El archivo *pom.xml* es un archivo que podemos manipular



directamente o bien podemos incluir dependencias gráficamente como veremos más adelante.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7     <groupId>com.mycompany</groupId>
8     <artifactId>conexionbd</artifactId>
9     <version>1.0-SNAPSHOT</version>
10    <packaging>jar</packaging>
11    <properties>
12        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
13        <maven.compiler.source>17</maven.compiler.source>
14        <maven.compiler.target>17</maven.compiler.target>
15        <exec.mainClass>com.mycompany.conexionbd.Conexionbd</exec.mainClass>
16    </properties>
17 </project>
```

Imagen 5. Archivo pom.xml por defecto (sin modificaciones)

Como se ve en la imagen existen etiquetas que tienen la funcionalidad de indicar por ejemplo la versión del sistema el formato de empaquetado (jar), algunas propiedades (properties) y falta la etiqueta *dependencies* que podemos agregar en el mismo nivel que la etiqueta properties (</properties>) y debajo del cierre de la misma.

Al igual que en un lenguaje de programación el código se tabula para incorporar bloques de código dentro de otros, los niveles o jerarquía de las etiquetas en XML también pueden reconocerse por el tabulado o espaciado. Por ejemplo, puede verse que la etiqueta *properties* está dentro de la etiqueta *project*.

Haciendo click con el botón derecho sobre la carpeta *dependencies* podemos agregar dependencias en la opción *add dependencies*. La primera vez que realice búsquedas, *Apache NetBeans* se conectará con el Central Repository y descargará el índice para las descargas. Este proceso puede demorar mucho tiempo dependiendo de su

velocidad de conexión. En la siguiente figura se muestra una búsqueda del controlador de bases de datos.

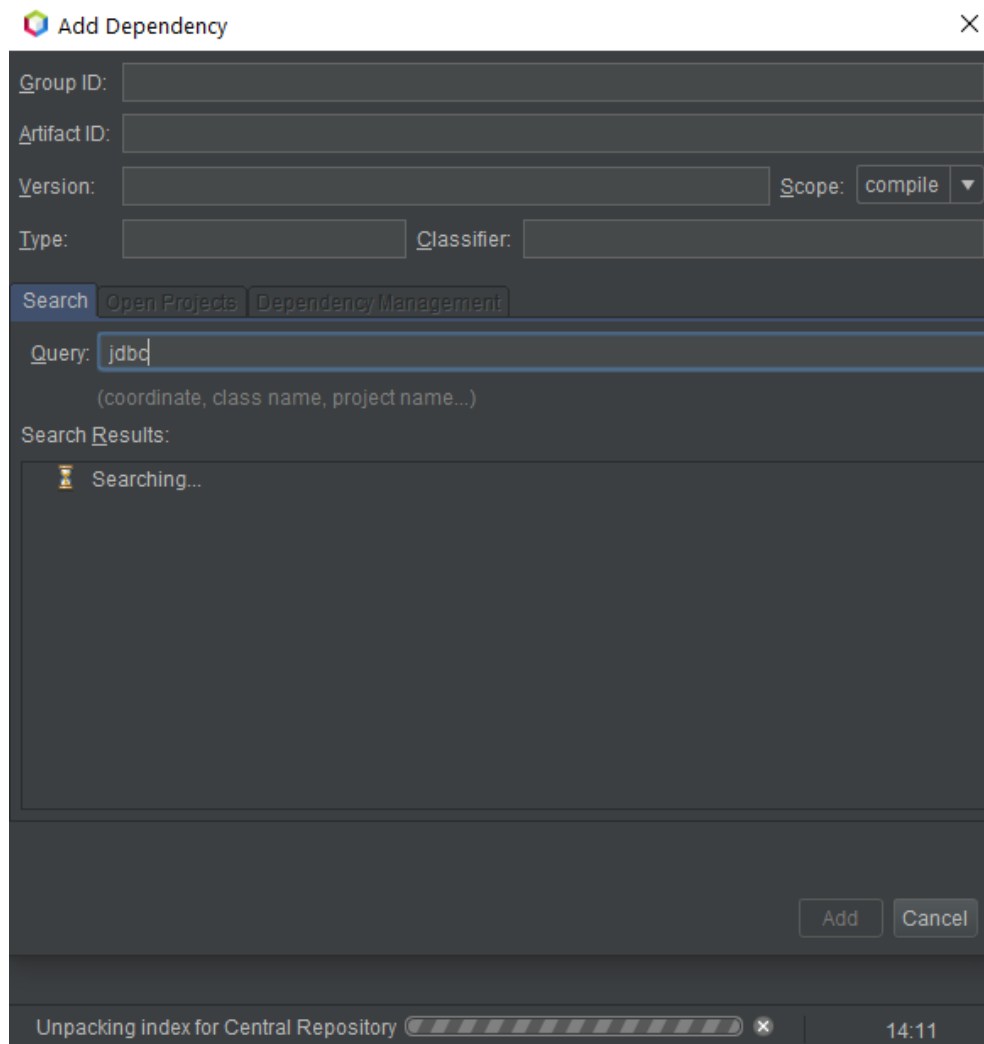
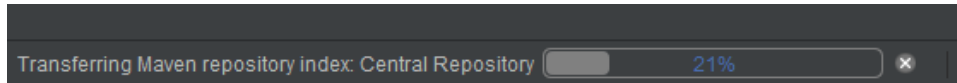


Imagen 6. Ventana de búsqueda de dependencias.

En la parte inferior de la imagen se puede ver que Netbeans está desempaquetando el índice para el repositorio central (*Unpacking index for Central Repository*).

Luego, NetBeans comenzará a hacer una transferencia del índice del repositorio como se muestra en la siguiente figura. Este proceso también suele tardar mucho, varias horas en algunos casos.



*Imagen 7. Transferring Maven Repository*

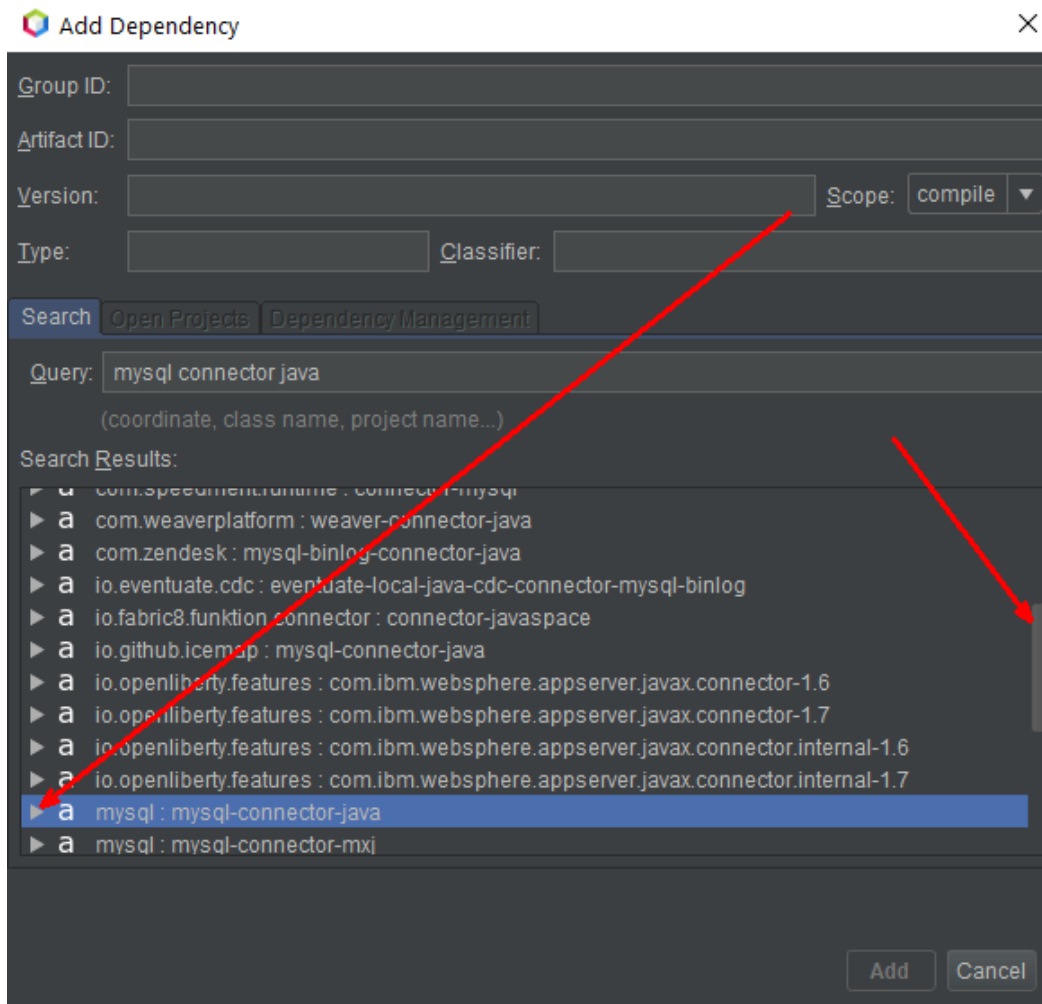
Lo que se recomienda, es realizar este primer proyecto con maven cuando la conexión de internet sea buena y haya poca gente usando la red, para que el proceso sea lo más rápido posible. En caso contrario, NetBeans podría tardar hasta 6 u 8 horas en hacer esto.

### Configuración del archivo pom.xml

Una vez que descargue y desempaque todos los archivos necesarios podremos por fin gestionar dependencias mediante el archivo pom.xml. A modo informativo, lo que se ha descargado e instalado ha quedado en un **directorio oculto** de su sistema a disposición de futuros proyectos. Por ejemplo, los directorios de almacenamiento para windows y linux respectivamente son los siguientes:

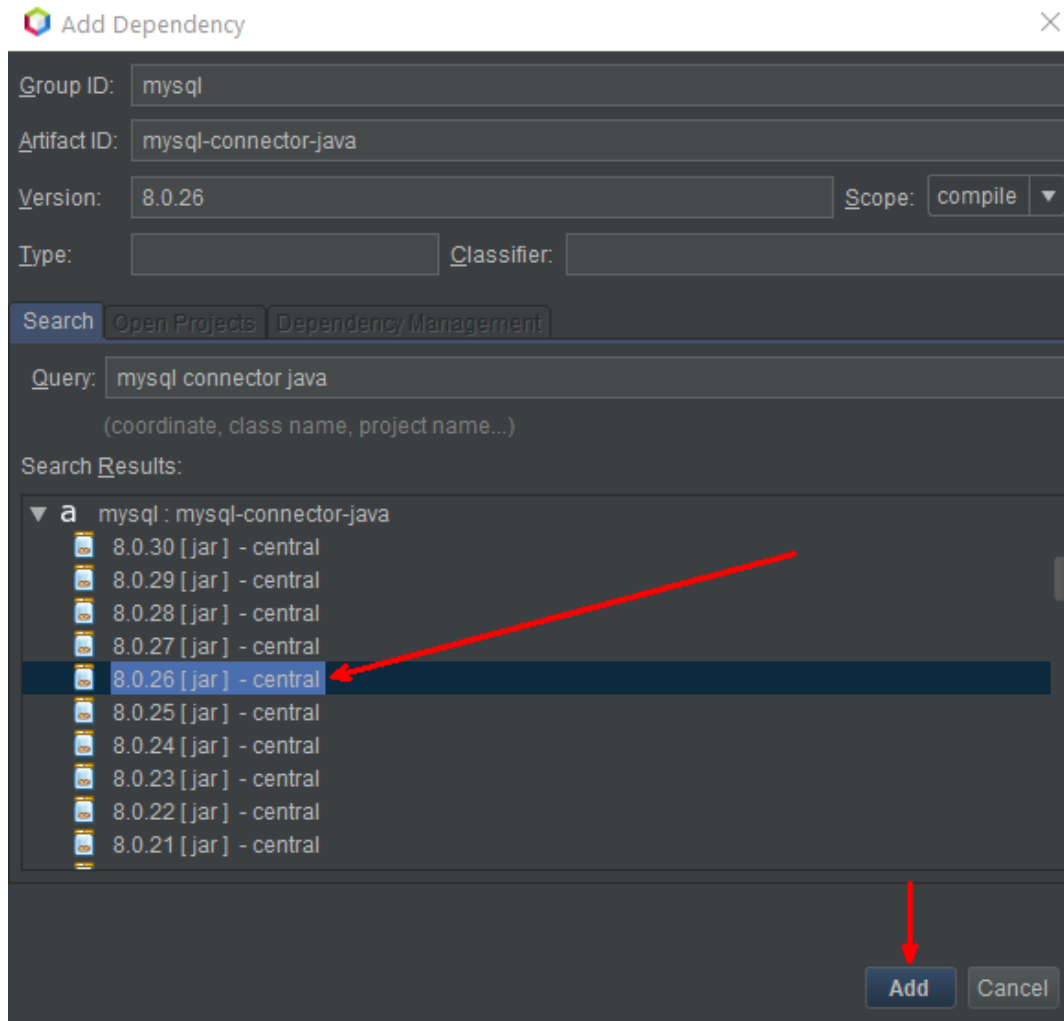
- Linux: **`/home/${usuario}/.m2/`**
- Windows: **`c:/Users/${usuario}/.m2/`**

Entonces, hacemos click derecho en la carpeta *dependencies* (imagen 4 de este capítulo) y luego add dependencies y buscamos por ejemplo “mysql connector java”, vamos a poder visualizar en el buscador varias opciones como se ve en la siguiente imagen.



*Imágen 8. Primera búsqueda del controlador de base de datos JDBC*

Desplazándonos hacia abajo en las opciones podremos encontrar “mysql:mysql-connector-java”, pero notemos que todavía no está habilitado el botón *add* por lo que debemos desplegar todas las opciones disponibles con la flecha señalada en la imagen anterior.



Imágen 9. Elección de la versión del controlador

Una vez desplegadas las opciones elegimos la versión que corresponda (en mi caso elijo 8.0.26) y podemos ahora agregar mediante el botón *add* y listo ya tenemos gestionada la dependencia para MySQL. De la misma manera podríamos gestionar dependencias para usar: Spring, Hibernate, Json entre algunas tecnologías disponibles. Solo tenemos que agregar la dependencia como ya está descrito y Maven se ocupará de descargar los archivos JAR para que el proyecto funcione de manera adecuada.

Una vez agregada la dependencia, el archivo pom.xml cambia como se ve en la siguiente figura.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5                          http://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7      <groupId>com.mycompany</groupId>
8      <artifactId>conexionbd</artifactId>
9      <version>1.0-SNAPSHOT</version>
10     <packaging>jar</packaging>
11     <dependencies>
12         <dependency>
13             <groupId>mysql</groupId>
14             <artifactId>mysql-connector-java</artifactId>
15             <version>8.0.26</version>
16         </dependency>
17     </dependencies>
18     <properties>
19         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
20         <maven.compiler.source>17</maven.compiler.source>
21         <maven.compiler.target>17</maven.compiler.target>
22         <exec.mainClass>com.mycompany.conexionbd.Conexionbd</exec.mainClass>
23     </properties>
24 </project>
```

Imagen 14. Etiquetas dependencies y dependency de pom.xml

### ¿Qué versión de MySQL tengo instalada?

Para saber qué versión de mysql tenemos instalada en nuestro sistema podemos dirigirnos, mediante el explorador de windows, a la carpeta "C:\Program Files\MySQL\MySQL Server 8.0\bin" y buscar el archivo *mysql(.exe)*. Hacemos click con el botón derecho y elegimos *propiedades*. Podremos ver la versión del producto como se muestra en la siguiente imagen. Si la instalación fue hecha con XAMPP la ruta puede ser: C:\xampp\mysql\bin.

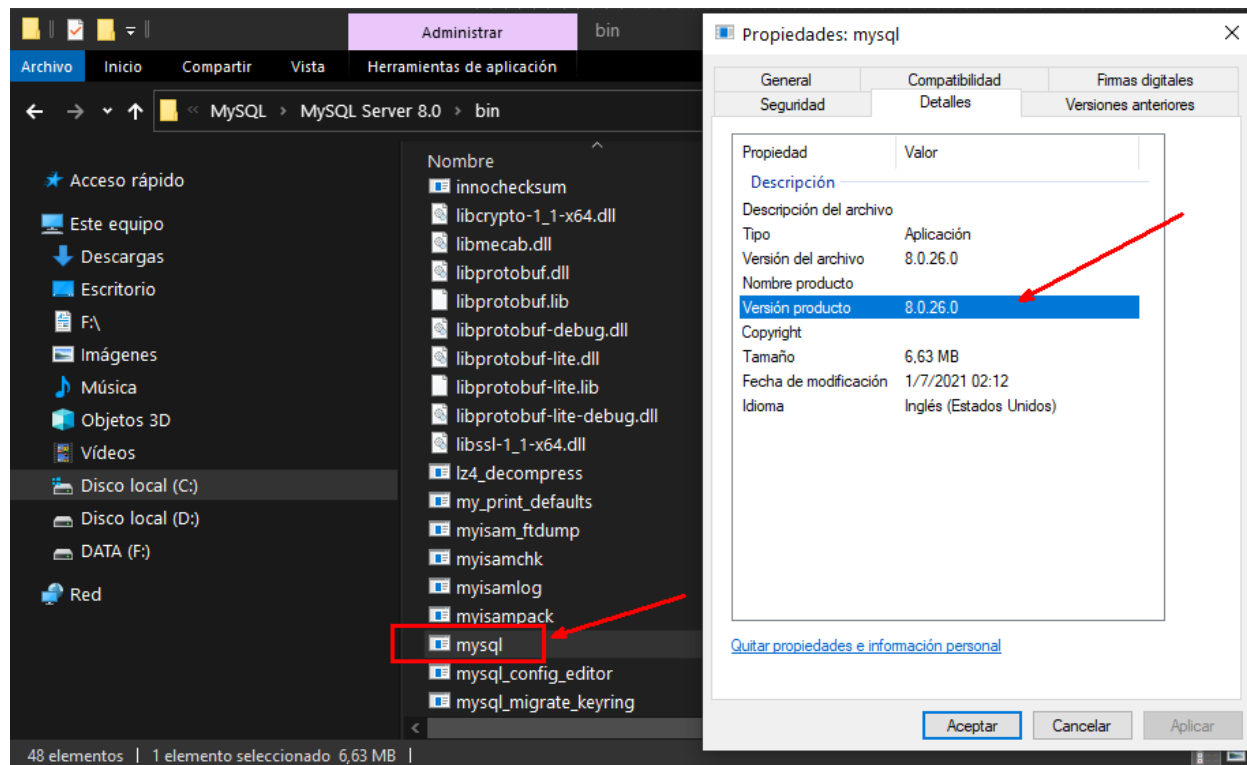


Imagen 10. Versión de MySQL.

Otra posibilidad es hacerlo mediante el *cmd* de windows. Si presionamos la combinación de teclas "windows + r" y luego escribimos **cmd** en la ventana podremos ejecutar el *command prompt* (o ventana para la ejecución de comandos de windows).

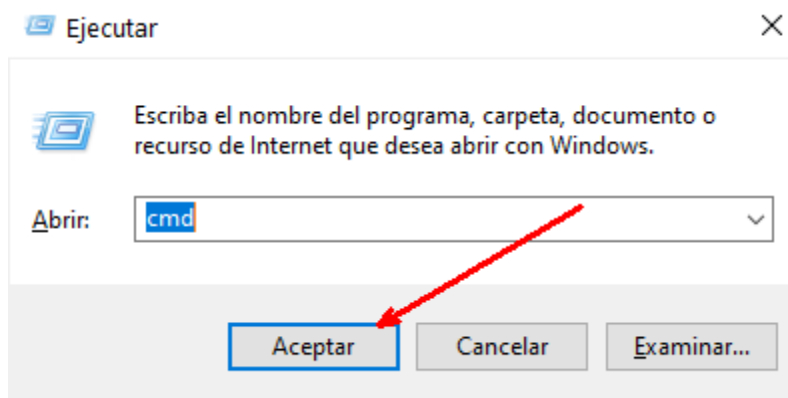
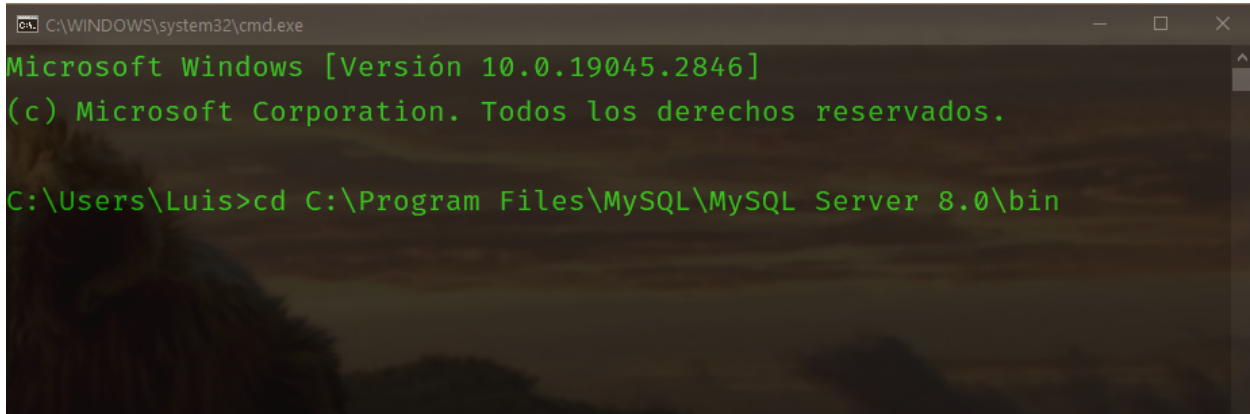


Imagen 11. Ventana ejecutar de Windows.

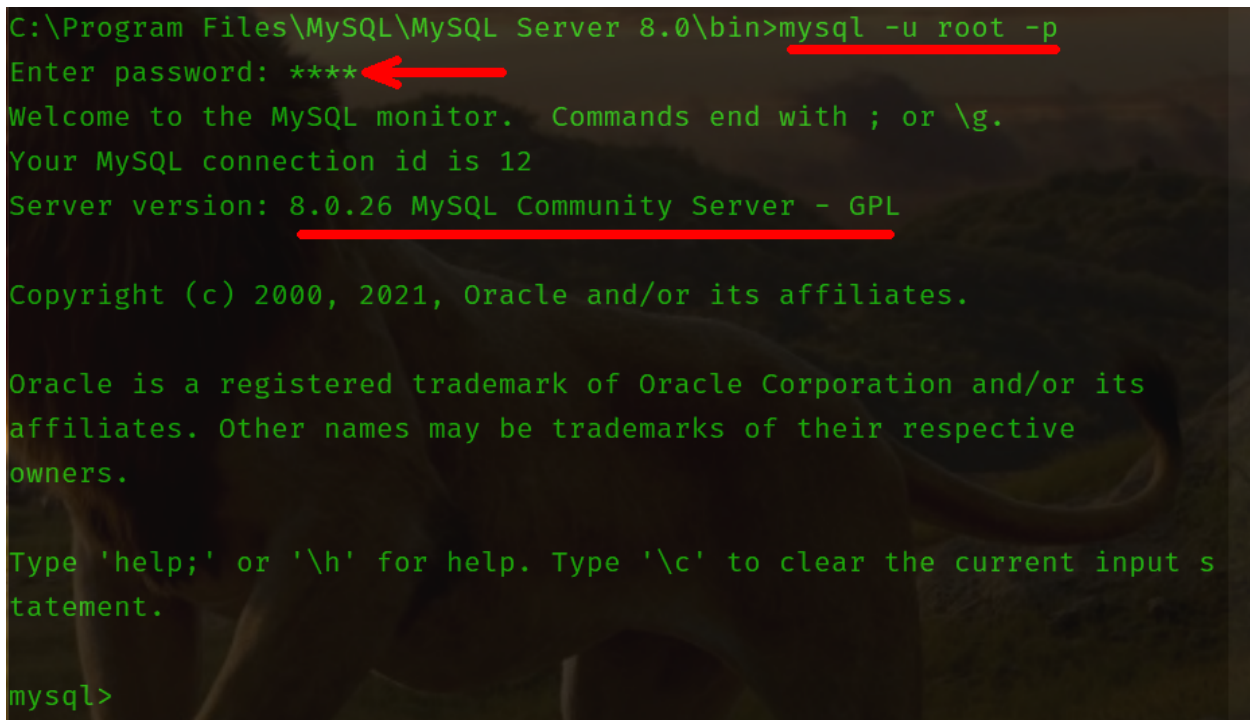
Luego, en el cmd pegamos la dirección donde está el ejecutable de mysql anteponiendo el comando `cd` (change directory) y luego un espacio: "`cd C:\Program Files\MySQL\MySQL Server 8.0\bin`" (o bien "`cd C:\xampp\mysql\bin`").



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2846]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>cd C:\Program Files\MySQL\MySQL Server 8.0\bin
```

*Imagen 12. Cambio de directorio mediante cd.*



```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input s
tatement.

mysql>
```

*Imagen 13. Mensaje de bienvenida del servidor.*



Luego, tecleamos `mysql -u root -p`. Se nos pedirá la misma contraseña que utilizamos para acceder a *MySQL Workbench* (en realidad es la contraseña que pusimos al instalar MySQL). Si instalamos MySQL con XAMPP es sin contraseña. Como se vé, en la imagen anterior, en el mensaje de bienvenida está la versión del servidor de mysql.

### Ejemplo de conexión a una base de datos

Para culminar este capítulo se mostrará como crear una base de datos sencilla y extraer información de ella.

Una vez que se ha descargado el controlador, pueden mirar que ahora la carpeta *dependecies* ya no está vacía. Creamos una base de datos llamada por ejemplo *tienda* desde *MySQL Workbench* (o *PHPMysqlAdmin* ambas sólo son interfaces para MySQL) ejecutando las siguientes sentencias:

```
1 • CREATE DATABASE tienda;  
2 • USE tienda;
```

Luego creamos dentro de *tienda* una tabla llamada *personas* con sus respectivos campos como se muestra a continuación.

Se deja un link donde se puede ver como hacer la creación con PHPMyAdmin:  
[https://youtu.be/SpX2g\\_I5Q\\_c](https://youtu.be/SpX2g_I5Q_c)

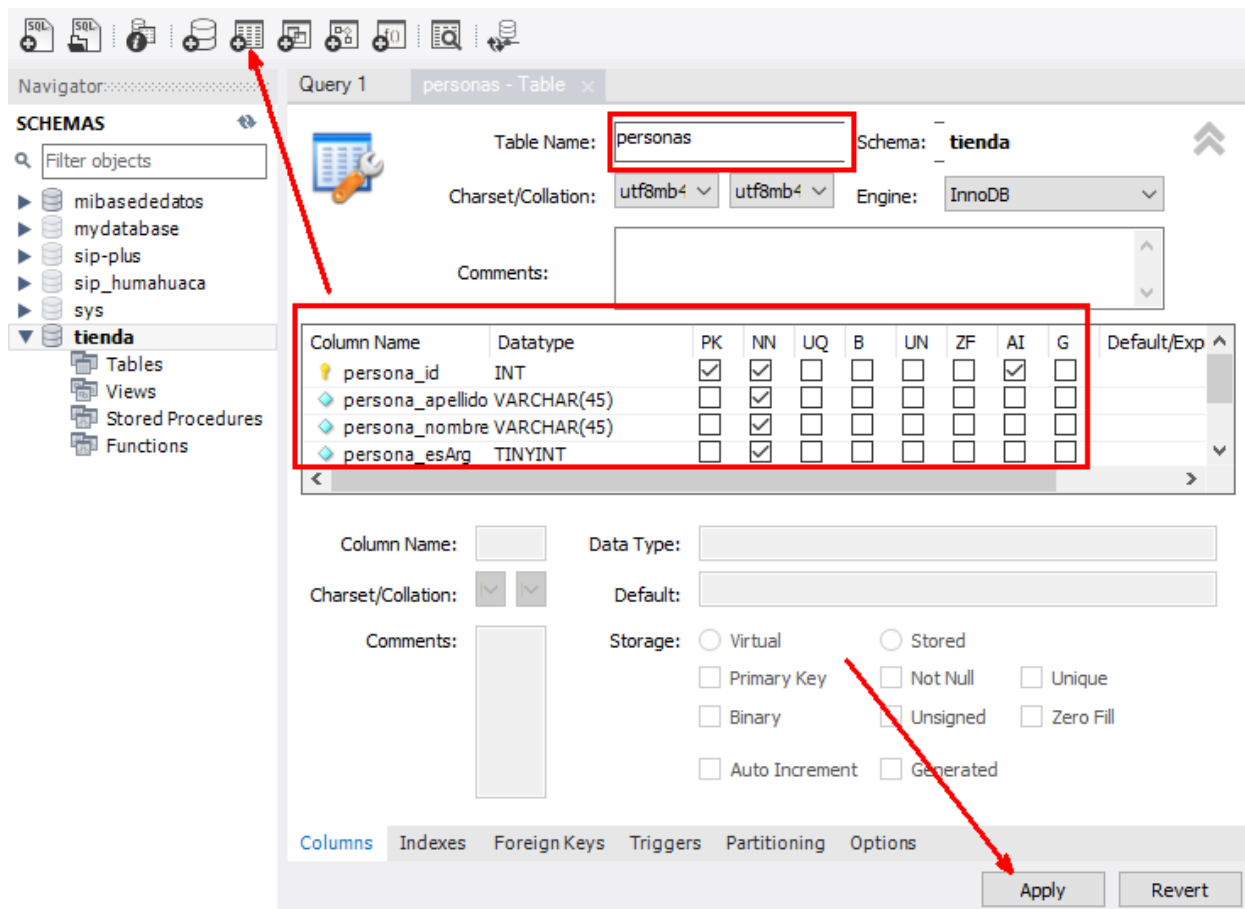


Imagen 15. Creación de una tabla mediante el asistente de Workbench.

Cuando presionamos *Apply*, el asistente mostrará las sentencias SQL que se ejecutarán en el servidor de bases de datos para la creación de la tabla, le damos click a *Apply* y luego a *Finish*.

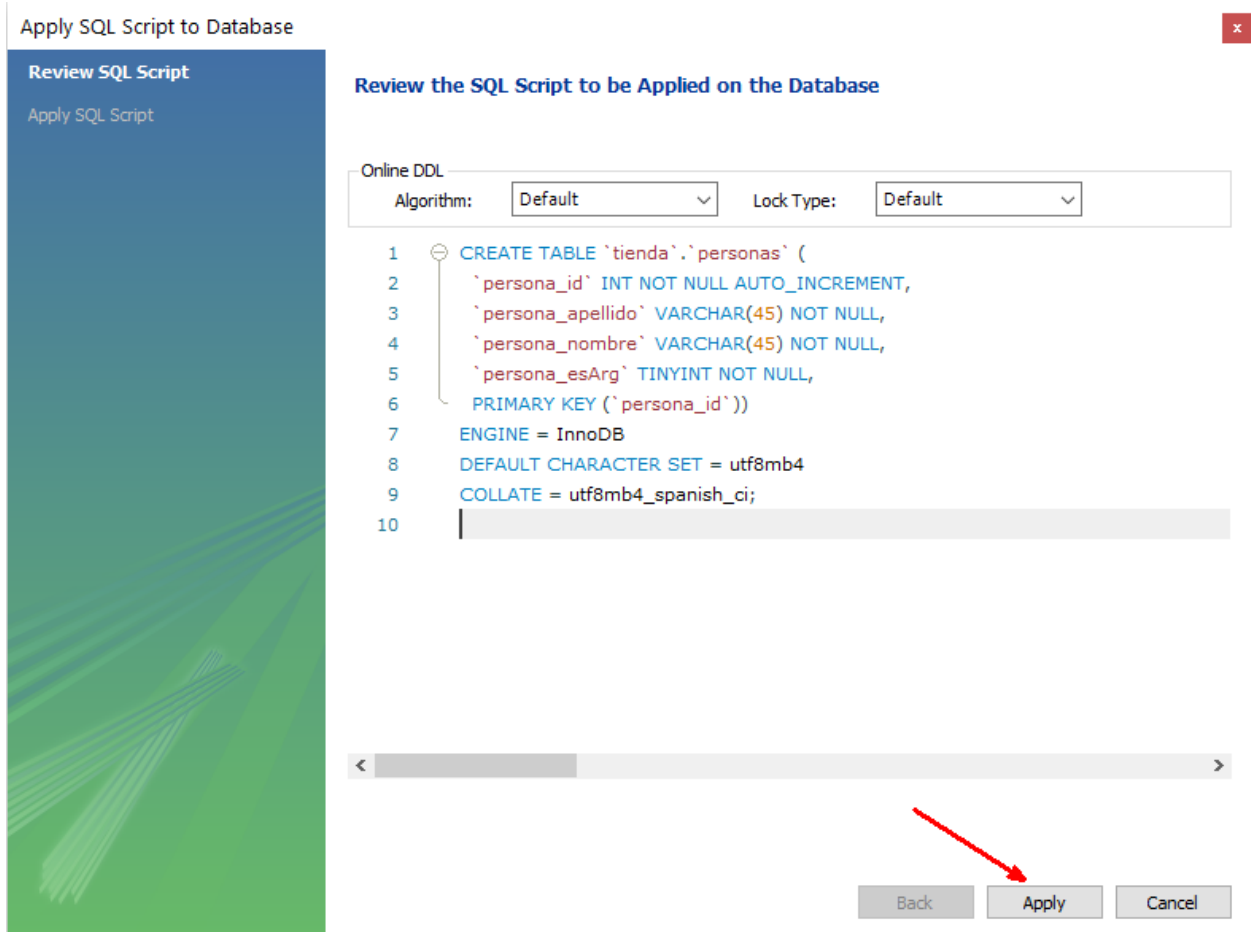


Imagen 16. Ejecución del código para la creación de la tabla personas.

Ejecutamos la siguiente sentencia (puede cambiar los datos que están entre comillas simples) para la inserción de un registro en la tabla *personas*:

```
insert into PERSONAS (persona_apellido, persona_nombre,  
persona_esArg) VALUES ('zapana', 'luis', true);
```

Y luego mostramos con un select el registro insertado, como se muestra a continuación.

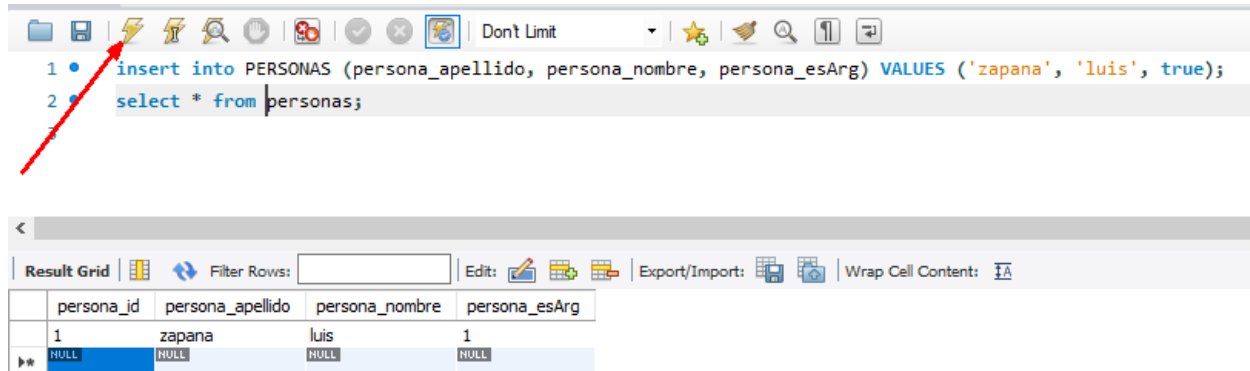


Imagen 17. Inserción y muestra de un registro de la tabla *personas*.

Si ha hecho todo correctamente debería poder ver un registro en su tabla.

Lo que haremos a continuación será extraer el registro pero desde Java. A fin de simplificar el ejemplo se pondrá todo el código en el método *main*, pero en una situación real la conexión y acceso a la base de datos debería estar declarada en otra clase. Para ello dentro del método *main* de un proyecto Maven insertamos el código que se muestra en la imagen 18.

Debemos conectarnos a la base de datos mediante la clase *Connection* y su método *.getConnection()*. Dicho método espera una cadena de conexión a la base de datos. En la línea 10 se crea la cadena de conexión a MySQL que contiene el número de puerto (en este caso yo he utilizado el puerto por defecto) y el nombre de la base de datos (tienda).

```
jdbc:mysql://localhost:3306/tienda
```

La conexión a la base de datos está hecha dentro de un bloque *try catch finally*, para que en caso de que la conexión falle se haga una gestión de la excepción. En la línea 33 se cierra la conexión.

```
1 package com.mycompany.conexionbd;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 public class Conexionbd {
9     public static void main(String[] args) throws SQLException {
10         final String url = "jdbc:mysql://localhost:3306/tienda";
11         Connection conn = null;
12         try {
13             conn = DriverManager.getConnection(url, "root", "root");
14             try (PreparedStatement stmt = conn.prepareStatement("SELECT * FROM PERSONAS")) {
15                 if (!conn.isClosed()) {
16                     System.out.println("MySQL working..");
17                 }
18                 try (ResultSet rs = stmt.executeQuery()) {
19                     while (rs.next()) {
20                         System.out.println("Nombre: " + rs.getString("persona_nombre"));
21                         System.out.println("Apellido: " + rs.getString("persona_apellido"));
22                         System.out.println("Es argentino: " + rs.getString("persona_esArg"));
23                     }
24                     rs.close();
25                 }
26                 stmt.close();
27             }
28         } catch (SQLException e) {
29             System.out.println("Exception: " + e.getMessage());
30         } finally {
31             try {
32                 if (conn != null) {
33                     conn.close();
34                 }
35             } catch (SQLException e) {
36                 System.out.println("Exception: " + e.getMessage());
37             }
38         }
39     }
40 }
```

Imagen 18. Código para la muestra de los registros de la tabla persona.

En la línea 13 se obtiene la conexión mediante la *cadena de conexión (url)* el *usuario root* y la *contraseña* que también es *root*. El bucle *while* de la línea 19 recorre y muestra todos los registros de la tabla (para este ejemplo hemos tenemos un solo registro).

Se deja a continuación el código en formato de texto para que el alumno pueda copiar si lo desea. Tenga cuidado de cambiar el nombre de la clase si Ud. creó una con un nombre distinto y con otras personalizaciones que haya hecho.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class Conexionbd {
    public static void main(String[] args) throws SQLException {
        final String url = "jdbc:mysql://localhost:3306/tienda";
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url, "root", "root");
            try (PreparedStatement stmt = conn.prepareStatement("SELECT
* FROM PERSONAS")) {
                if (!conn.isClosed()) {
                    System.out.println("MySQL working..");
                }
                try (ResultSet rs = stmt.executeQuery()) {
                    while (rs.next()) {
                        System.out.println("Nombre: " +
rs.getString("persona_nombre"));
                        System.out.println("Apellido: " +
rs.getString("persona_apellido"));
                        System.out.println("Es argentino: " +
rs.getString("persona_esArg"));
                    }
                    rs.close();
                }
                stmt.close();
            }
        } catch (SQLException e) {
            System.out.println("Exception: " + e.getMessage());
        } finally {
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
```



```
e.getMessage() );  
        }  
    }  
}
```

System.out.println("Exception: " +