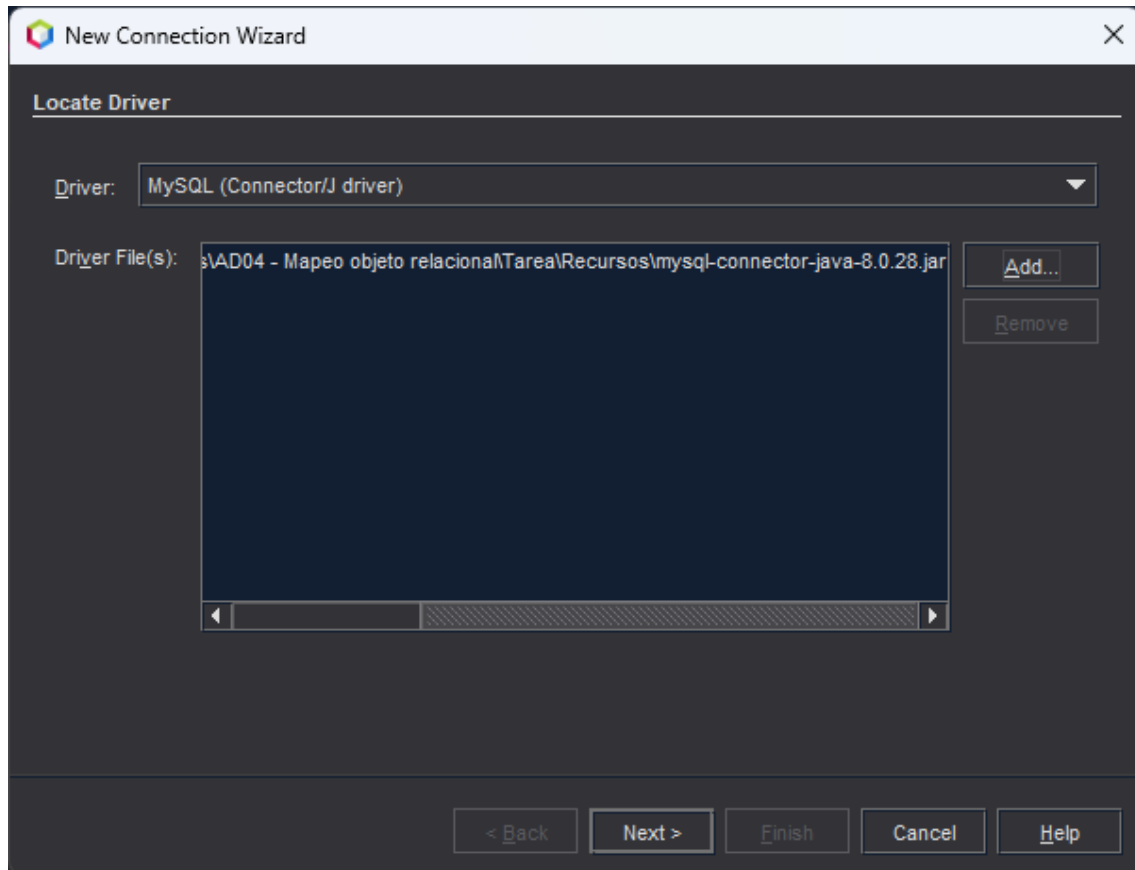
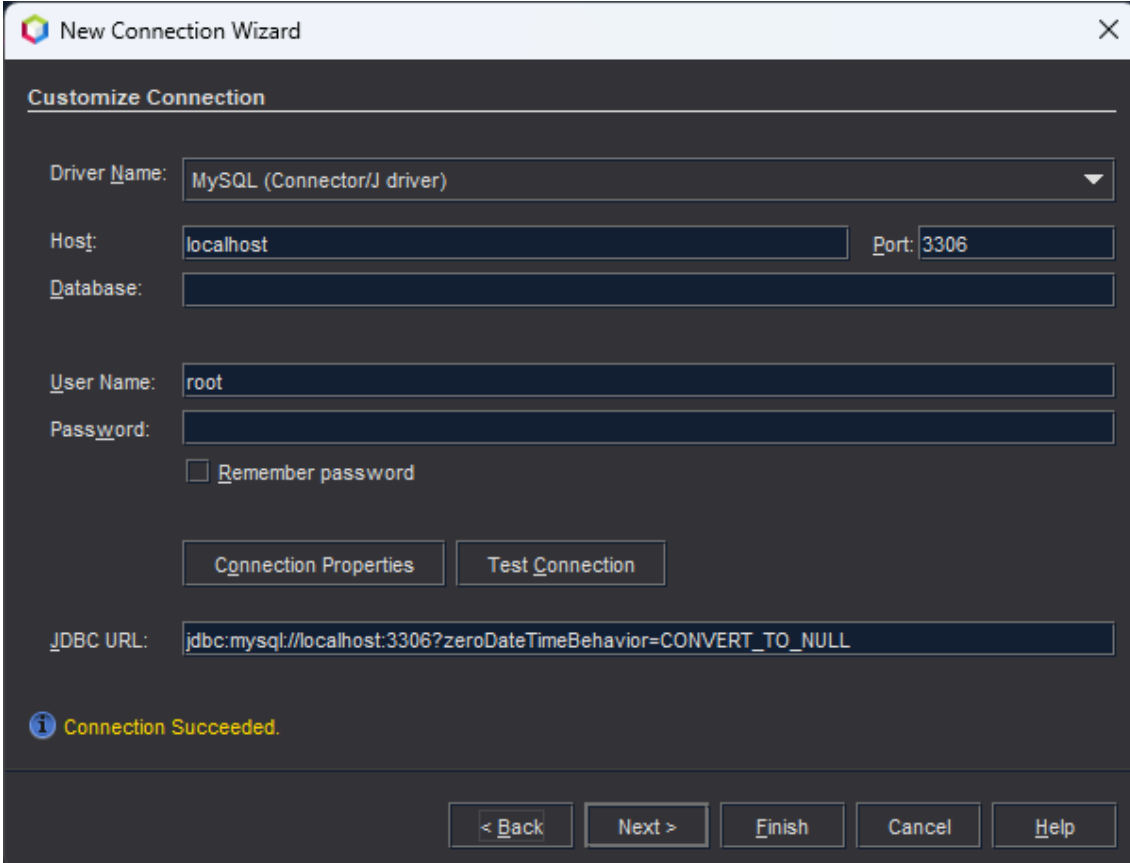


1. Crea la base de datos.

Para conectarnos con la base de datos, desde NetBeans nos vamos a la pestaña de servicios, “Services”. En el apartado de bases de datos, pulsamos con el botón derecho y seleccionamos “New connection...”.



Seleccionamos el tipo de controlador “MySql” y añadimos el archivo .jar del conector. En nuestro caso hemos utilizado la versión 8.0.28, que aprovechando que más adelante vamos a necesitar su dependencia Maven, hemos [descargado directamente desde su repositorio](#).



New Connection Wizard

Customize Connection

Driver Name: MySQL (Connector/J driver)

Host: localhost Port: 3306

Database:


User Name: root

Password:

☐ Remember password

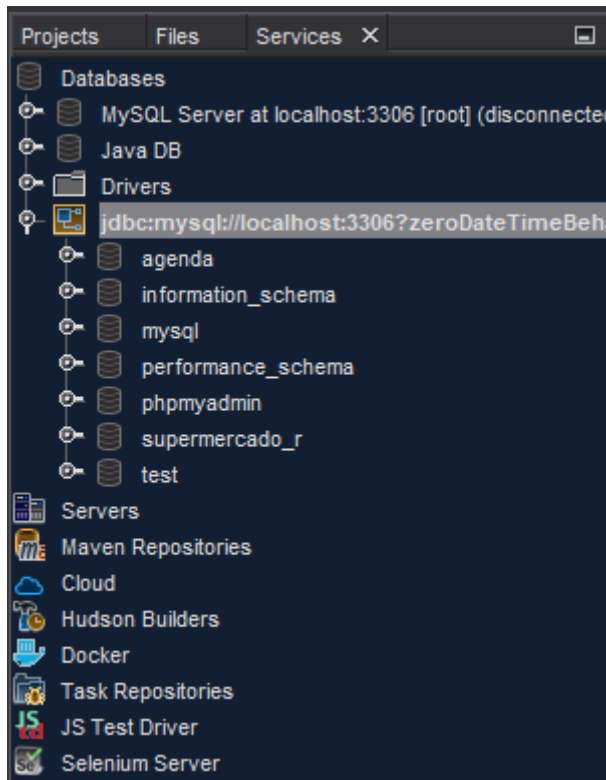
Connection Properties Test Connection

JDBC URL: jdbc:mysql://localhost:3306?zeroDateTimeBehavior=CONVERT_TO_NULL

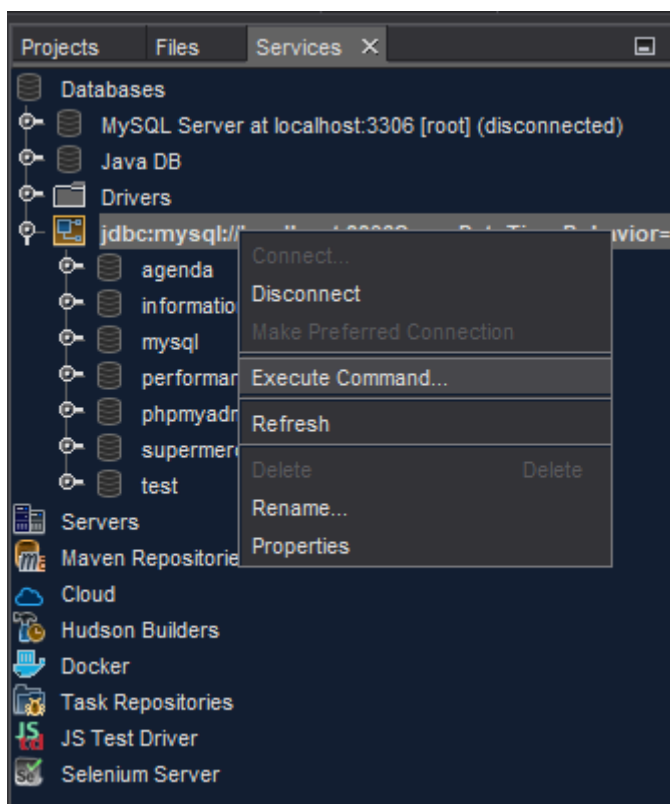
 Connection Succeeded.

< Back Next > Finish Cancel Help

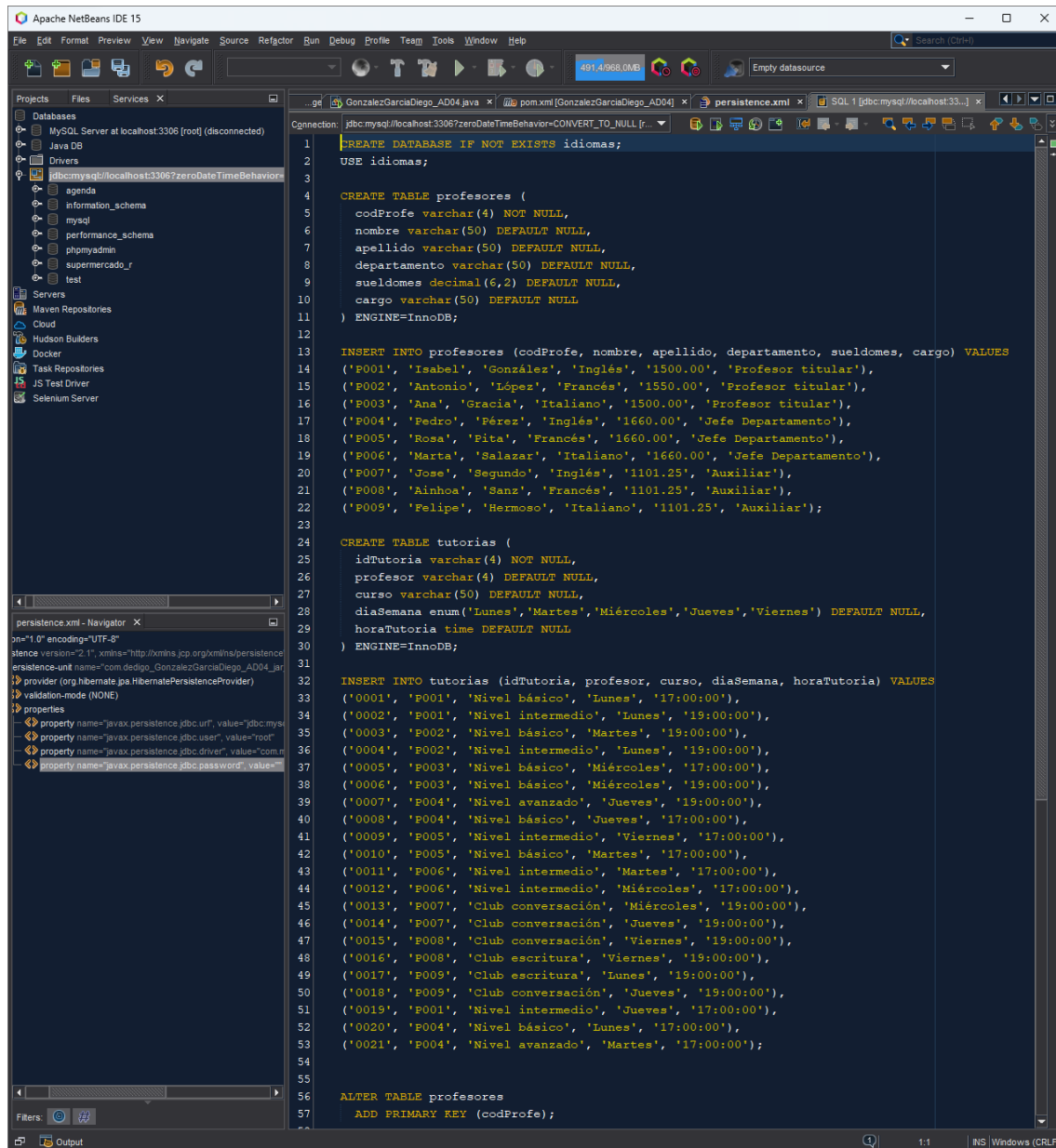
Indicamos los datos de la base de datos a la cual nos queremos conectar. Como nosotros todavía no la tenemos creada, comprobamos el conector sin nombrar ninguna base de datos pulsando sobre "Test_Connection", la cual nos indica que es satisfactoria, por lo que el conector funciona correctamente.



Una vez pulsado sobre finalizar, si desplegamos el conector que acabamos de añadir, vemos todas las bases de datos que tenemos creadas en nuestro sistema.

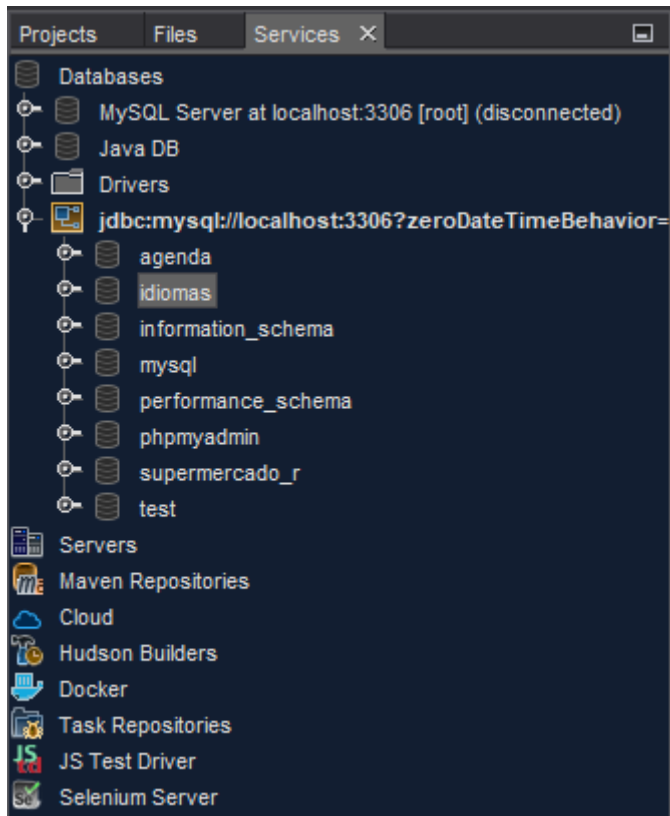


A continuación, pulsamos con el botón derecho sobre la conexión y seleccionamos “Execute Command”.

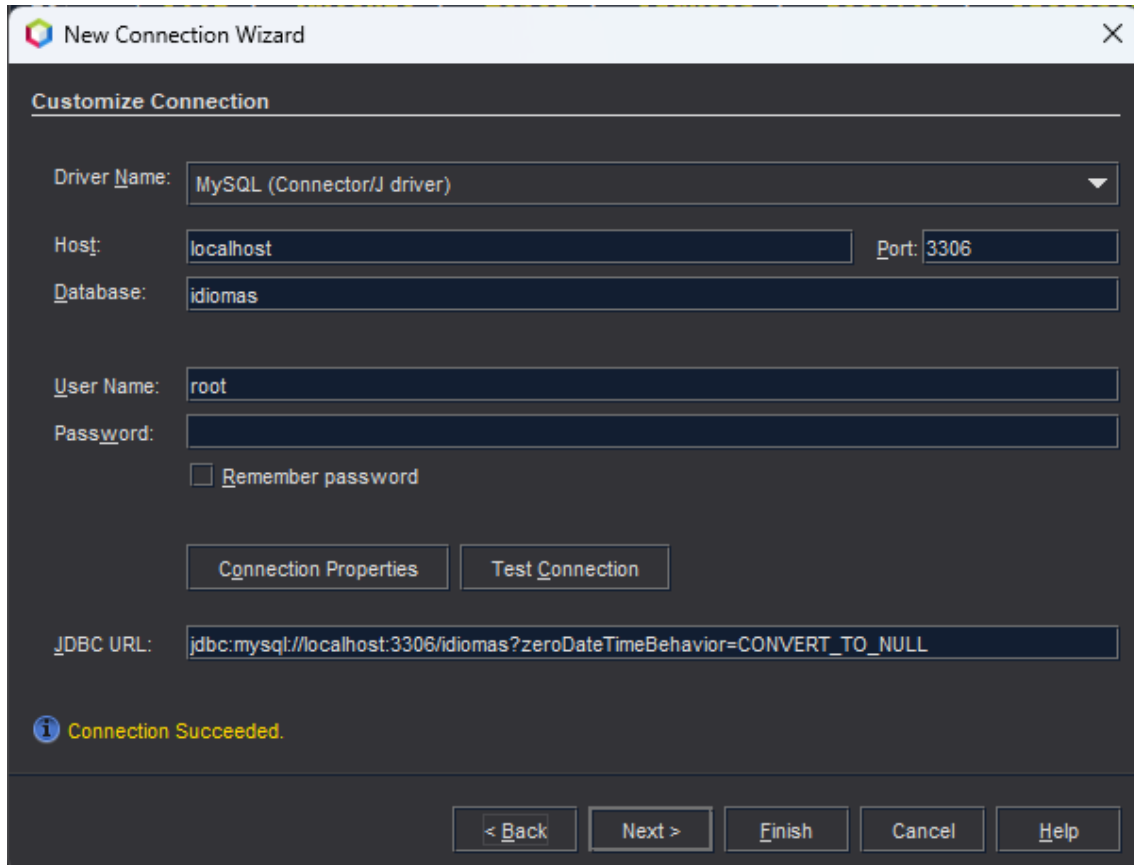


```
1 CREATE DATABASE IF NOT EXISTS idiomas;
2 USE idiomas;
3
4 CREATE TABLE profesores (
5     codProfe varchar(4) NOT NULL,
6     nombre varchar(50) DEFAULT NULL,
7     apellido varchar(50) DEFAULT NULL,
8     departamento varchar(50) DEFAULT NULL,
9     sueldomes decimal(6,2) DEFAULT NULL,
10    cargo varchar(50) DEFAULT NULL
11 ) ENGINE=InnoDB;
12
13 INSERT INTO profesores (codProfe, nombre, apellido, departamento, sueldomes, cargo) VALUES
14 ('P001', 'Isabel', 'González', 'Inglés', '1500.00', 'Profesor titular'),
15 ('P002', 'Antonio', 'López', 'Francés', '1550.00', 'Profesor titular'),
16 ('P003', 'Ana', 'Gracia', 'Italiano', '1500.00', 'Profesor titular'),
17 ('P004', 'Pedro', 'Pérez', 'Inglés', '1660.00', 'Jefe Departamento'),
18 ('P005', 'Rosa', 'Pita', 'Francés', '1660.00', 'Jefe Departamento'),
19 ('P006', 'Marta', 'Salazar', 'Italiano', '1660.00', 'Jefe Departamento'),
20 ('P007', 'Jose', 'Segundo', 'Inglés', '1101.25', 'Auxiliar'),
21 ('P008', 'Ainhoa', 'Sanz', 'Francés', '1101.25', 'Auxiliar'),
22 ('P009', 'Felipe', 'Hermoso', 'Italiano', '1101.25', 'Auxiliar');
23
24 CREATE TABLE tutorias (
25     idTutoria varchar(4) NOT NULL,
26     profesor varchar(4) DEFAULT NULL,
27     curso varchar(50) DEFAULT NULL,
28     diaSemana enum('Lunes','Martes','Miércoles','Jueves','Viernes') DEFAULT NULL,
29     horaTutoria time DEFAULT NULL
30 ) ENGINE=InnoDB;
31
32 INSERT INTO tutorias (idTutoria, profesor, curso, diaSemana, horaTutoria) VALUES
33 ('0001', 'P001', 'Nivel básico', 'Lunes', '17:00:00'),
34 ('0002', 'P001', 'Nivel intermedio', 'Lunes', '19:00:00'),
35 ('0003', 'P002', 'Nivel básico', 'Martes', '19:00:00'),
36 ('0004', 'P002', 'Nivel intermedio', 'Lunes', '19:00:00'),
37 ('0005', 'P003', 'Nivel básico', 'Miércoles', '17:00:00'),
38 ('0006', 'P003', 'Nivel básico', 'Miércoles', '19:00:00'),
39 ('0007', 'P004', 'Nivel avanzado', 'Jueves', '19:00:00'),
40 ('0008', 'P004', 'Nivel básico', 'Jueves', '17:00:00'),
41 ('0009', 'P005', 'Nivel intermedio', 'Viernes', '17:00:00'),
42 ('0010', 'P005', 'Nivel básico', 'Martes', '17:00:00'),
43 ('0011', 'P006', 'Nivel intermedio', 'Martes', '17:00:00'),
44 ('0012', 'P006', 'Nivel intermedio', 'Miércoles', '17:00:00'),
45 ('0013', 'P007', 'Club conversación', 'Miércoles', '19:00:00'),
46 ('0014', 'P007', 'Club conversación', 'Jueves', '19:00:00'),
47 ('0015', 'P008', 'Club conversación', 'Viernes', '19:00:00'),
48 ('0016', 'P008', 'Club escritura', 'Viernes', '19:00:00'),
49 ('0017', 'P009', 'Club escritura', 'Lunes', '19:00:00'),
50 ('0018', 'P009', 'Club conversación', 'Jueves', '19:00:00'),
51 ('0019', 'P001', 'Nivel intermedio', 'Jueves', '17:00:00'),
52 ('0020', 'P004', 'Nivel básico', 'Lunes', '17:00:00'),
53 ('0021', 'P004', 'Nivel avanzado', 'Martes', '17:00:00');
54
55
56 ALTER TABLE profesores
57     ADD PRIMARY KEY (codProfe);
```

En la pantalla que se nos abre, escribimos/pegamos el script para crear la base de datos y pulsamos sobre la opción “Run SQL”(icono amarillo con un “play” verde) que se encuentra a la derecha de la conexión.



Ya tenemos nuestra BD creada y por lo tanto, vamos a crear una conexión a ella para utilizarla en la tarea más adelante.

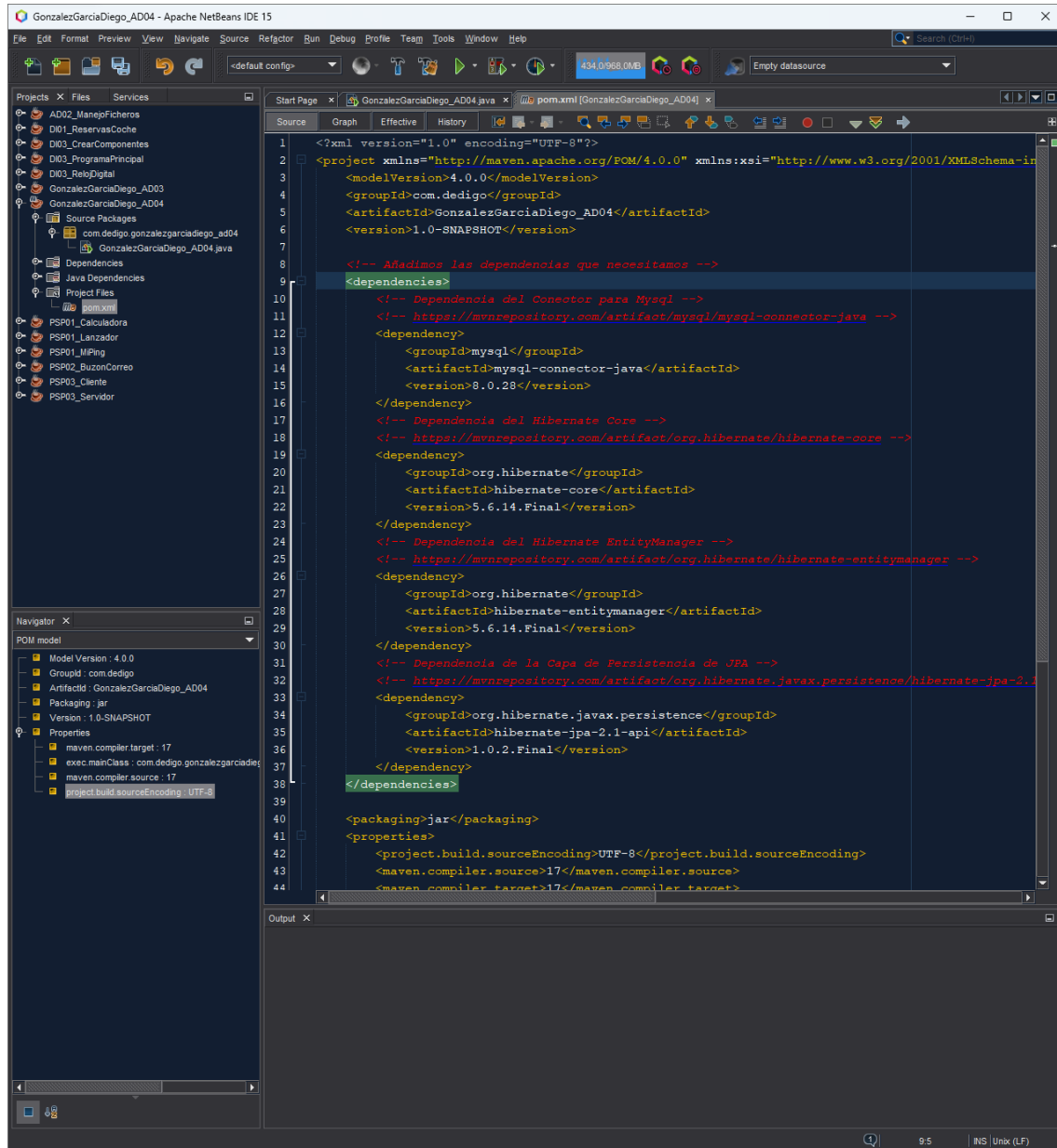


Realizamos la conexión de la misma forma que previamente, pero en este caso indicamos la BD.

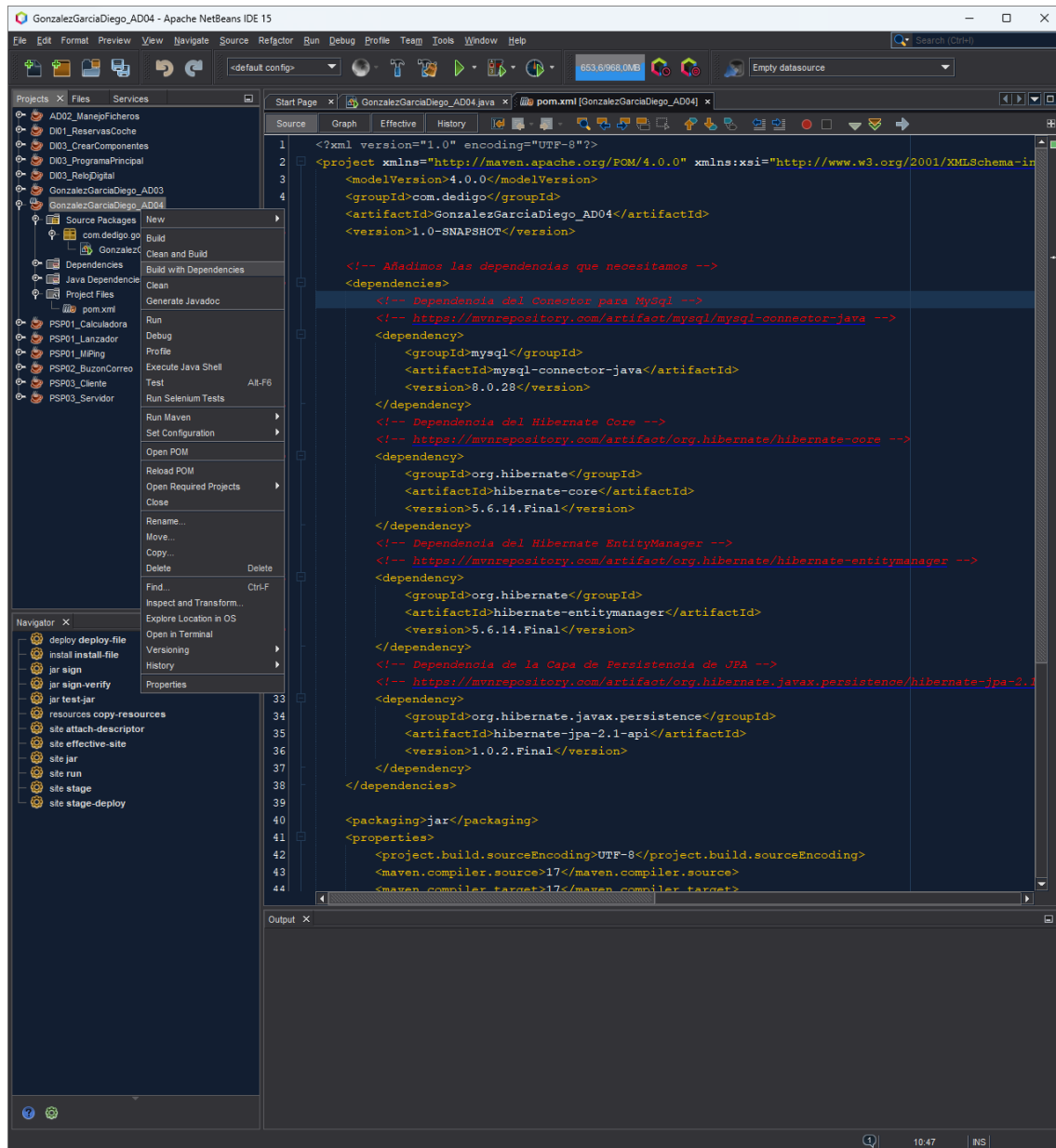
2. Configura y crea la ORM Hibernate.

a. Creación del proyecto e inclusión de dependencias.

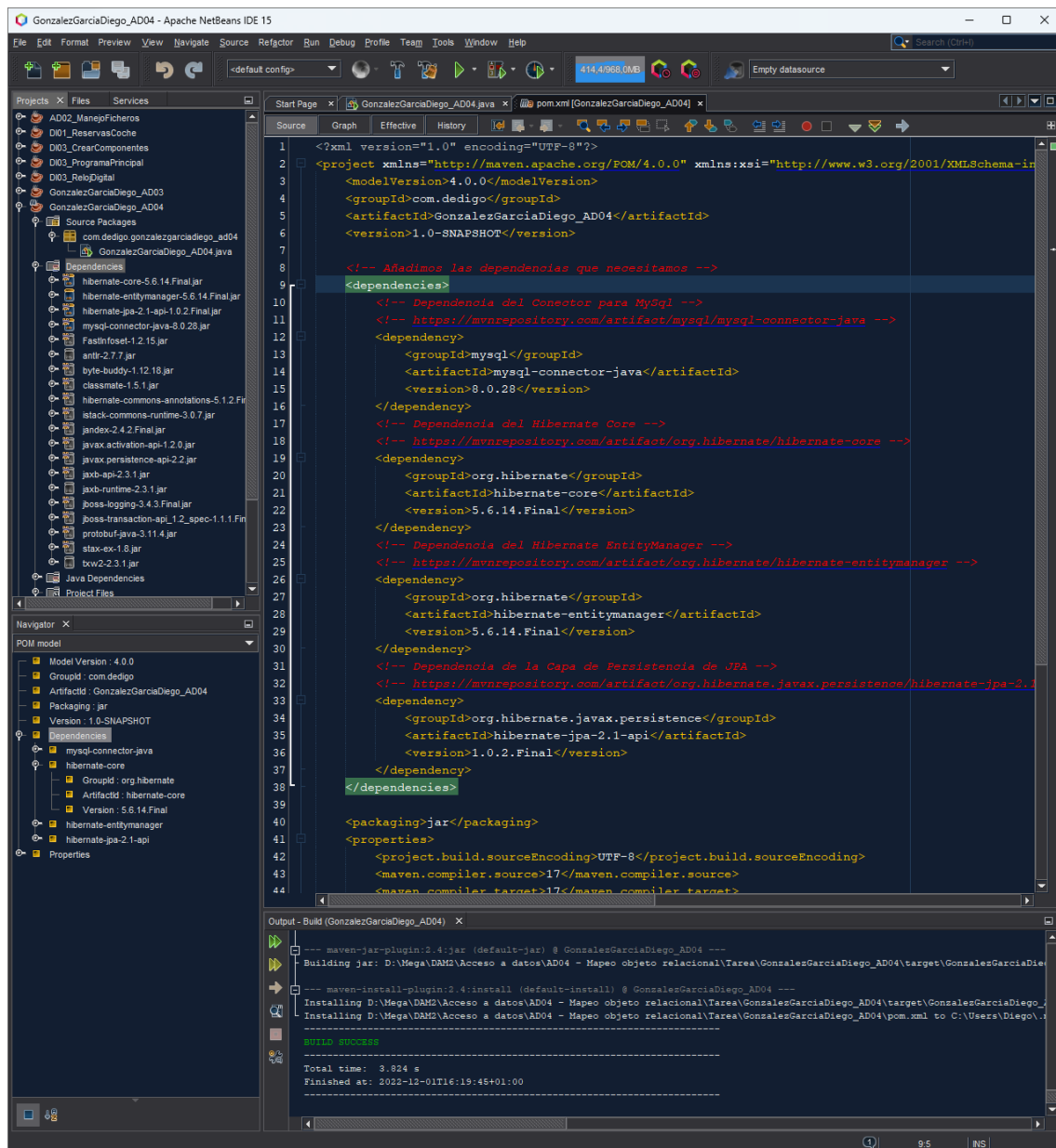
Comenzamos creando un proyecto con Maven. Una vez creado, nos dirigimos al archivo “pom.xml”, en el cual vamos a añadir las dependencias que nos van a hacer falta.



Como podemos observar en la imagen, hemos añadido todas las dependencias que nos van a hacer falta (Conector MySql, Hibernate Core, Hibernate EntityManager, Hibernate JPA) las cuales hemos buscado desde el [repositorio de Maven](https://mvnrepository.com/). Importante utilizar la misma versión de Hibernate Core y de Hibernate EntityManager, en nuestro caso la versión 5.6.14.Final.



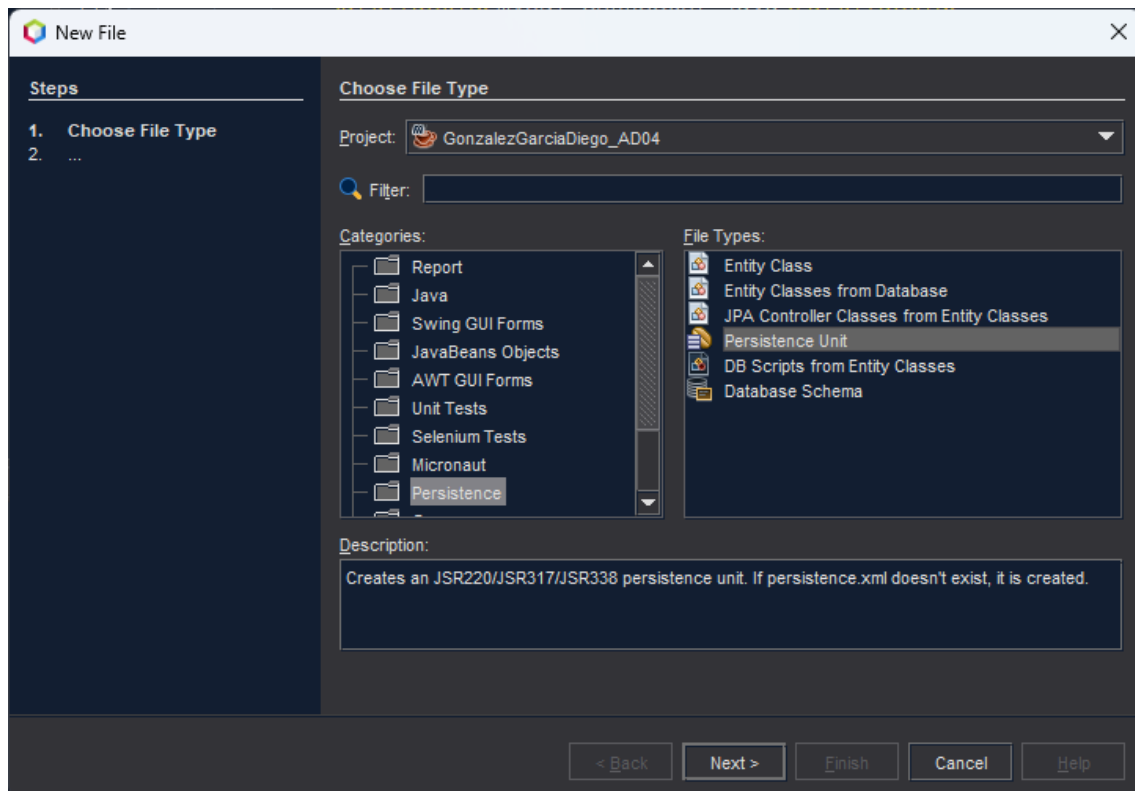
Una vez que hemos añadido todas las dependencias, pulsamos con el botón derecho sobre nuestro proyecto y seleccionamos “Build with Dependencies”. Tras unos segundos, ya disponemos de todas nuestras dependencias descargadas.



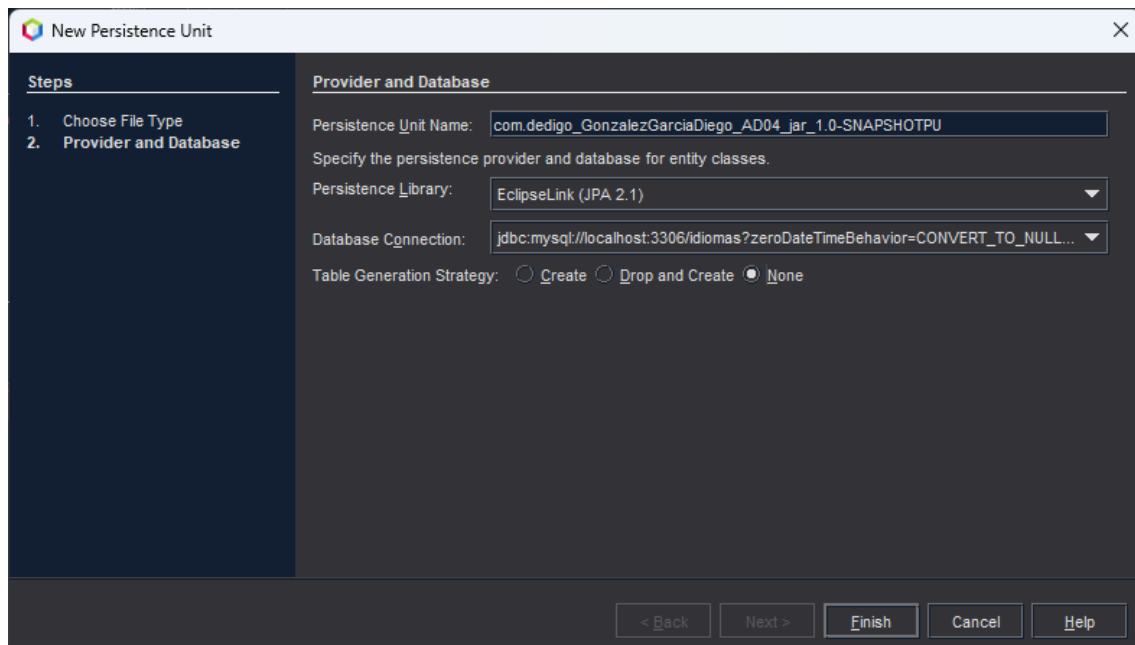
Si abrimos la carpeta de dependencias de nuestro proyecto, vemos que ya se han añadido todos los ficheros que le hemos indicado en el “pom.xml”.

b. Creación del archivo de persistencia.

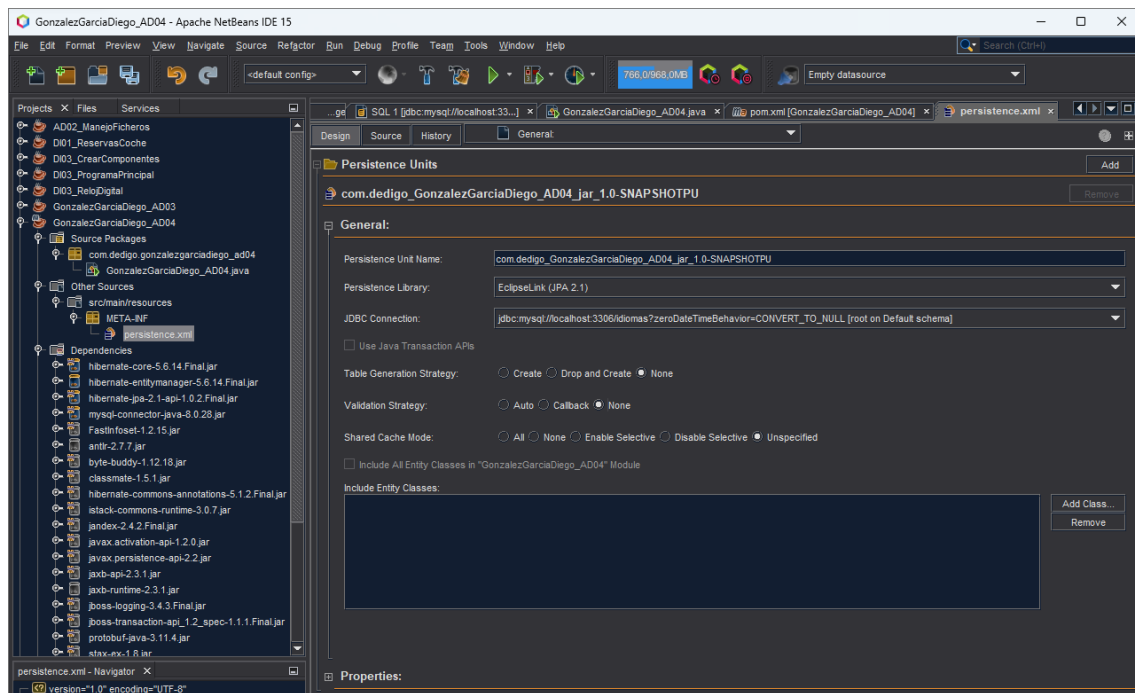
Para ello, pulsamos con el botón derecho sobre nuestro proyecto, “New/Other...”.



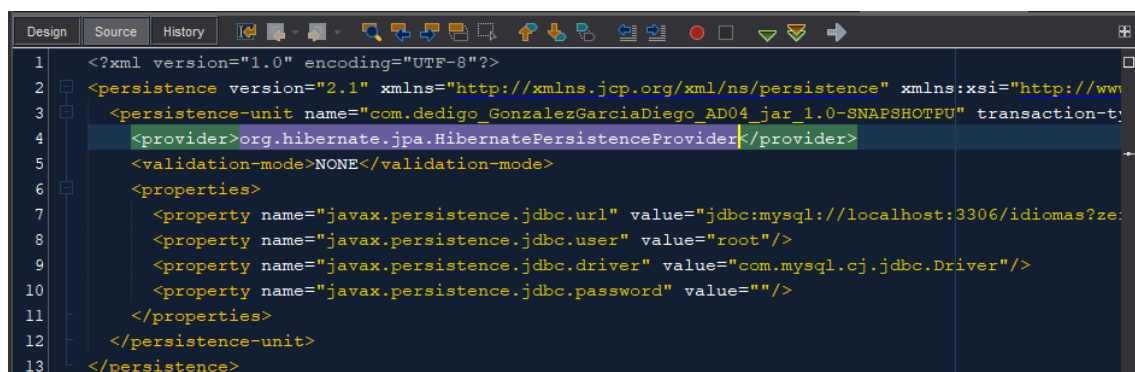
En la categoría de persistencia, seleccionamos “Persistence Unit”.



Dejamos la librería por defecto, “EclipseLink (JPA 2.1)”, que más adelante modificaremos, seleccionamos el conector de nuestra base de datos, el cual hemos creado en el primer apartado de esta tarea, y como estrategia de generación de tabla indicamos “None”.



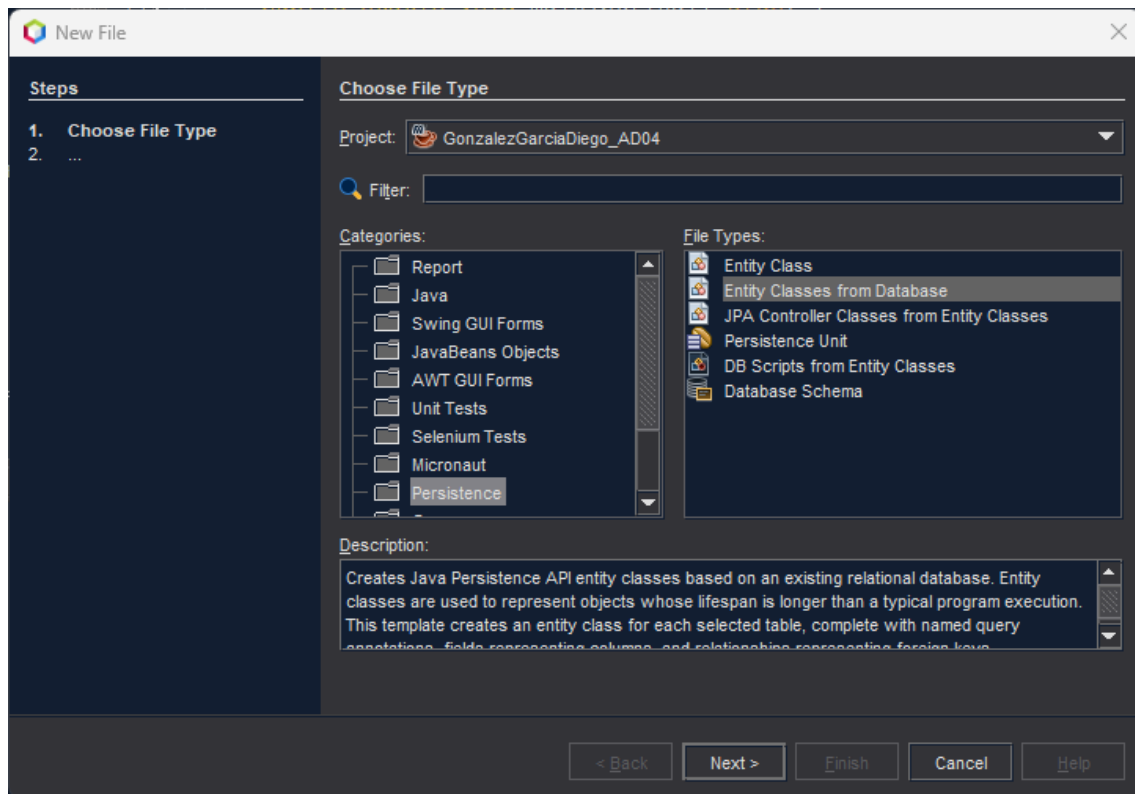
Tras pulsar sobre finalizar, en la siguiente pantalla indicamos que la estrategia de validación también es “None”.



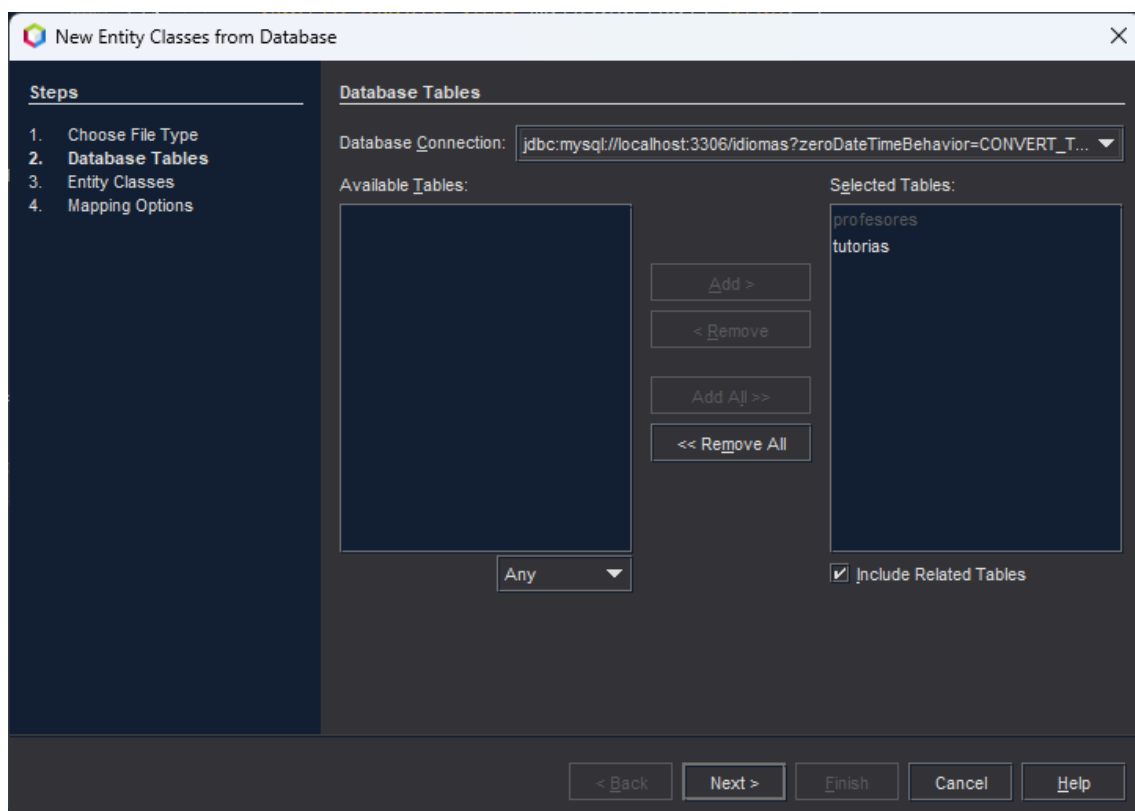
A continuación, desde la pestaña “Source” del archivo “persistence.xml”, modificamos el proveedor dejándolo de la siguiente forma: “org.hibernate.jpa.HibernatePersistenceProvider”.

c. Creación de POJOs

De nuevo, pulsamos con el botón derecho sobre nuestro proyecto, “New/Other...”.



En la categoría de persistencia, seleccionamos “Entity Classes from Database”. De esta forma se nos van a generar automáticamente las clases de nuestra base de datos.



Añadimos todas las tablas que tenemos en la base de datos. Importante tener seleccionada la conexión con la base de datos que queremos utilizar, en nuestro caso “idiomas”.

The screenshot shows the 'New Entity Classes from Database' dialog box, specifically the 'Entity Classes' step. The 'Steps' panel on the left lists: 1. Choose File Type, 2. Database Tables, 3. Entity Classes (selected), and 4. Mapping Options. The main area is titled 'Entity Classes' and contains the instruction 'Specify the names and the location of the entity classes.' Below this is a table for 'Class Names':

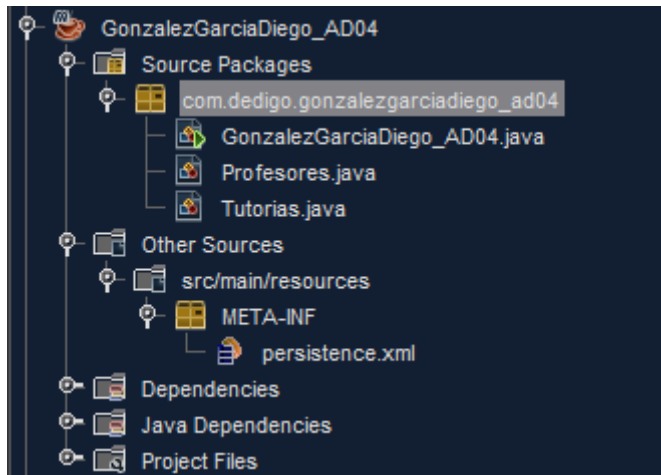
Database Table	Class Name	Generation Type
profesores	Profesores	New
tutorias	Tutorias	New

Below the table is a '+' button. Further down are fields for 'Project' (GonzalezGarciaDiego_AD04), 'Location' (Source Packages), and 'Package' (com.dedigo.gonzalezgarciadiego_ad04). At the bottom are three checkboxes: ☒ Generate Named Query Annotations for Persistent Fields, ☒ Generate JAXB Annotations, and ☐ Generate MappedSuperclasses instead of Entities. At the very bottom are buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

La siguiente pantalla la dejamos por defecto.

The screenshot shows the 'New Entity Classes from Database' dialog box, specifically the 'Mapping Options' step. The 'Steps' panel on the left lists: 1. Choose File Type, 2. Database Tables, 3. Entity Classes, and 4. Mapping Options (selected). The main area is titled 'Mapping Options' and contains the instruction 'Specify the default mapping options.' Below this are two dropdown menus: 'Association Fetch' (set to 'default') and 'Collection Type' (set to 'java.util.List'). At the bottom are five checkboxes: ☒ Fully Qualified Database Table Names, ☐ Attributes for Regenerating Tables, ☒ Use Column Names in Relationships, ☐ Use Defaults if Possible, and ☐ Generate Fields for Unresolved Relationships. At the very bottom are buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

En esta pantalla indicamos que los tipos de datos que queremos que se devuelvan sean de tipo "List" y marcamos la opción "Fully Qualified Database Table Names".



Tras unos segundos, vemos que en el proyecto ya nos aparecen las Clases con el nombre de las tablas que contenía nuestra base de datos.

3. Realiza una inserción y un borrado sobre la tabla tutorías.

a. Método para mostrar el listado de las tutorías

```
/**
 * <strong>Método que muestra todas las tutorías que hay en la base de
 * datos</strong>
 */
static void mostrarTutorias() {
    entityManager.clear();
    //Actualizo la lista de tutorias
    listTutorias = entityManager.createNamedQuery( string:"Tutorias.findAll").getResultList();
    //Defino un patrón de formato para la salida de datos por consola, como un bloque de texto
    patron = ""
        %-10s%-25s%-15s%-15s%-10s
        -----"";
    //Muestro la cabecera de los datos utilizando el patrón creado
    System.out.println( x:String.format( format:patron,
        args:"CÓDIGO",
        args:"CURSO",
        args:"DÍA",
        args:"HORA",
        args:"PROFESOR"
    ));
    //Recorro la lista de tutorias mientras voy obteniendo y mostrando sus datos por pantalla
    for (Tutorias tutorias : listTutorias) {
        //Utilizo el patrón creado y añado sus respectivos datos
        resultado = String.format( format:patron,
            args:tutorias.getIdTutoria(),
            args:tutorias.getCurso(),
            args:tutorias.getDiaSemana(),
            args:tutorias.getHoraTutoria(),
            args:tutorias.getProfesor().getCodProfe()
        );
        //Muestro la información por pantalla
        System.out.println( x:resultado);
    }
}
```

b. Método que inserta una tutoría en la base de datos

```
/**
 * <strong>Método que inserta una tutoría con id nº22</strong>
 */
static void insertarTutoria() {
    System.out.println(":\n\nInsertamos una tutoría nueva con el id nº 0022\n\n");
    Profesores profesor = null;
    //Obtenemos el profesor cuyo código es P001
    try {
        profesor = (Profesores) entityManager.createNamedQuery( string:"Profesores.findByCodProfe")
            .setParameter( string:"codProfe", o:"P001").getSingleResult();
    } catch (NoResultException ex) {
        System.err.println(":\tEl profesor no existe\n");
    }

    //Creamos una nueva tutoría con el id 0022 y añadimos sus datos
    Tutorias tutoria = new Tutorias( idTutoria:"0022");
    tutoria.setCurso( curso:"Nivel básico");
    tutoria.setDiaSemana( diaSemana:"Lunes");
    tutoria.setHoraTutoria( horaTutoria:hora(h:13, m:15));
    tutoria.setProfesor(profesor);

    try {
        //Comenzamos una transacción con la base de datos
        entityManager.getTransaction().begin();
        //Añadimos la tutoría
        entityManager.persist(o:tutoria);
        //Finalizamos la transacción
        entityManager.getTransaction().commit();
    } catch (EntityExistsException ex) {
        System.err.println(":\tYa existe una tutoría con el mismo identificador\n");
    }
}
```

c. Método que borra una tutoría de la base de datos

```
/**
 * <strong>Método que borra la tutoría con id nº20</strong>
 */
static void borrarTutoria() {
    System.out.println(":\n\nBorramos la tutoría con el id nº 0020\n\n");
    Tutorias tutoria = null;
    //Obtenemos el profesor cuyo código es P001
    try {
        tutoria = (Tutorias) entityManager.createNamedQuery( string:"Tutorias.findByIdTutoria")
            .setParameter( string:"idTutoria", o:"0020").getSingleResult();
        //Comenzamos una transacción con la base de datos
        entityManager.getTransaction().begin();
        //Borramos la tutoría
        entityManager.remove(o:tutoria);
        //Finalizamos la transacción
        entityManager.getTransaction().commit();
    } catch (NoResultException ex) {
        System.err.println(":\tLa tutoría no existe\n");
    }
}
```

4. Obtener un listado sobre las tablas profesores y tutorías que visualice codProfe, nombre, apellido, departamento, diaSemana y horaTutoria.

Todos los datos del profesor para cada tutoría, ya se encuentran incluidos en el objeto profesor que utiliza el POJO, por lo que no es necesario cruzar datos entre las tablas para mostrar toda su información.

```
/**
 * <strong>Método que resuelve el punta número 4 de la tarea</strong><br>
 * Obtener un listado sobre las tablas profesores y tutorías que visualice
 * codProfe, nombre, apellido, departamento, diaSemana y horaTutoria.
 */
static void mostrarProfesorTutoria() {
    System.out.println( x:"\n\nEJERCICIO NÚMERO 4:\n\n");
    //Actualizo la lista de tutorías
    listTutorias = entityManager.createNamedQuery( string:"Tutorias.findAll").getResultList();
    //Defino un patrón de formato para la salida de datos por consola, como un bloque de texto
    patron = ""
        %-10s%-15s%-15s%-15s%-15s%-10s
        -----";
    //Muestro la cabecera de los datos utilizando el patrón creado
    System.out.println( x:String.format( format:patron,
        args:"CÓDIGO",
        args:"NOMBRE",
        args:"APELLIDO",
        args:"DEPARTAMENTO",
        args:"DÍA",
        args:"HORA"
    ));
    //Recorro la lista de tutorías mientras voy obteniendo y mostrando sus datos por pantalla
    for (Tutorias tutorias : listTutorias) {
        //Utilizo el patrón creado y añado sus respectivos datos
        resultado = String.format( format:patron,
            args:tutorias.getProfesor().getCodProfe(),
            args:tutorias.getProfesor().getNombre(),
            args:tutorias.getProfesor().getApellido(),
            args:tutorias.getProfesor().getDepartamento(),
            args:tutorias.getDiaSemana(),
            args:tutorias.getHoraTutoria()
        );
        //Muestro la información por pantalla
        System.out.println( x:resultado);
    }
}
```