

Unidad 3

Mapeo objeto relacional

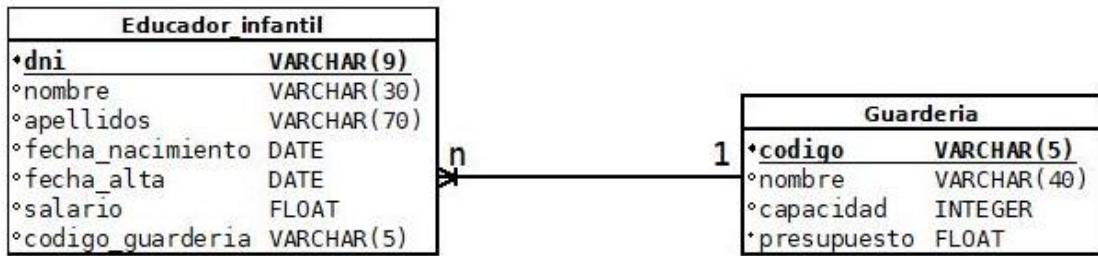
Tarea AD03

Manejo de mapeo objeto-relacional (ORM) usando Hibernate

Contenido

EJERCICIO 1:.....	2
a) Lanzar el script SQL de creación del esquema de la base de datos (Educacion_Infantil) y asegurar que se crean ambas tablas y se insertan los registros correspondientes. Puedes usar, bien un cliente gráfico (MySQL Workbench) o bien un cliente en modo texto (Línea de comandos MySQL).	3
b) Desarrollar un proyecto en Netbeans con nombre AD03_Apellido1_Nombre y configurar Hibernate para poder realizar el mapeo de la base de datos creada en el apartado anterior. Debes detallar todo lo realizado referente al mapeo (creación del fichero de configuración hibernate.cfg.xml, generación del fichero de persistencia, etc.)......	8
EJERCICIO 2:.....	15
a) Añadir nuevas guarderías y nuevos educadores infantiles a la base de datos.	16
b) Eliminar guarderías y educadores infantiles de la base de datos. Si una guardería tiene educadores asignados no se podrá eliminar la guardería.....	24
c) Mostrar el listado de todas las guarderías.	32
d) Mostrar un listado con todos los educadores infantiles (ordenados por apellidos) de una guardería indicada por el usuario mediante su nombre.....	35
e) Mostrar un listado con todos los educadores infantiles que ganan más de una determinada cantidad indicada por el usuario (ordenados por salario de mayor a menor).	38
f) Modificar el salario de un educador a una determinada cantidad (el usuario debe facilitar dni del educador y nuevo salario).....	41

La base de datos centrosEducacionInfantil tiene la siguiente estructura:

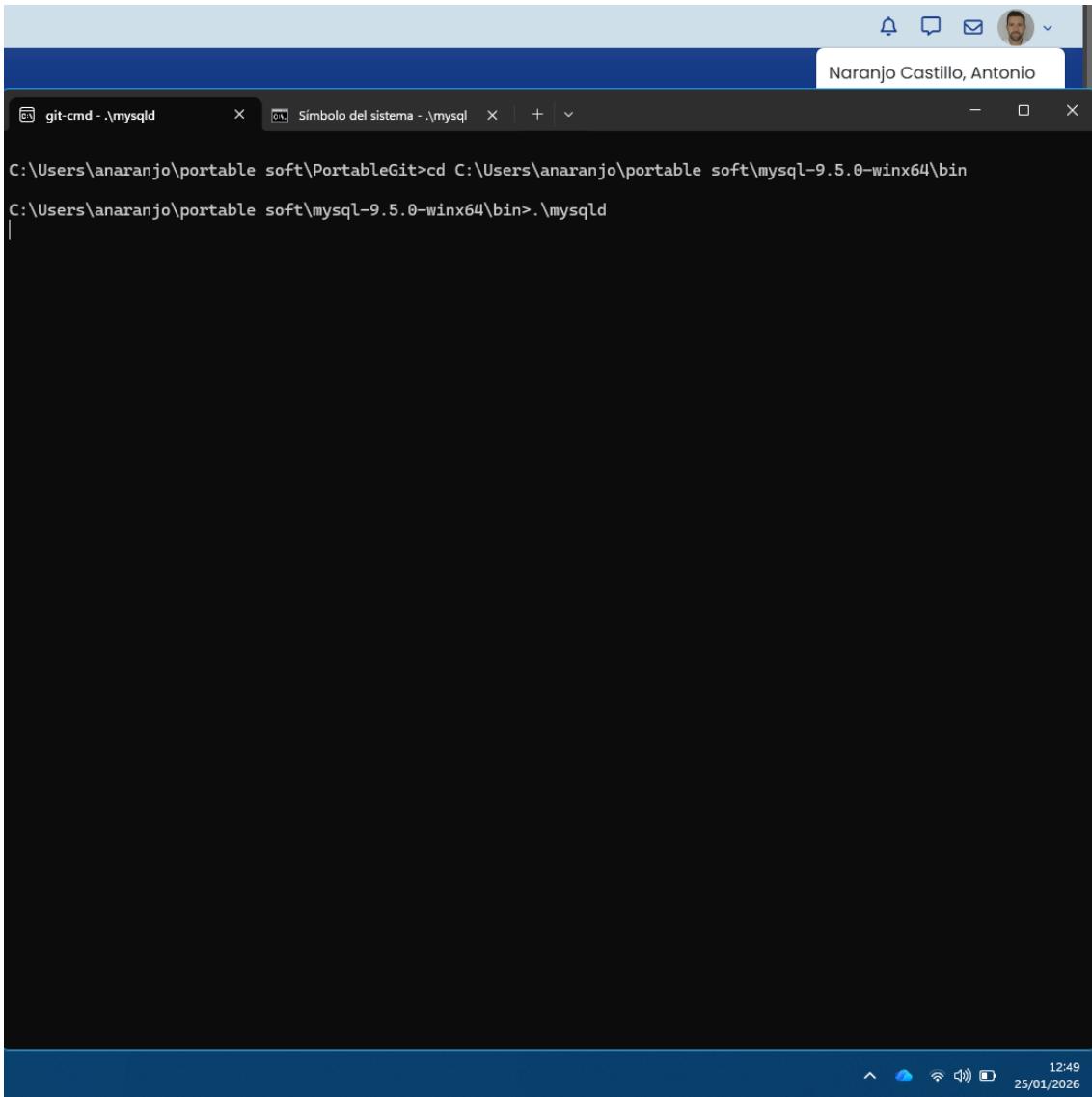


A partir de dicha base de datos, realiza los siguientes ejercicios:

EJERCICIO 1:

Elabora un documento donde documentes de forma detallada mediante explicaciones y capturas de pantalla los pasos realizados para llevar a cabo las siguientes tareas:

a) Lanzar el script SQL de creación del esquema de la base de datos (*Educacion_Infantil*) y asegurar que se crean ambas tablas y se insertan los registros correspondientes. Puedes usar, bien un cliente gráfico (MySQL Workbench) o bien un cliente en modo texto (Línea de comandos MySQL).



The screenshot shows a Windows desktop environment. At the top, there is a blue header bar with icons for notifications, messages, and user profile. Below it is a dark blue taskbar. On the taskbar, there are two open windows: one titled "git-cmd - .\mysqld" and another titled "Símbolo del sistema - .\mysql". The "git-cmd" window has a white background and contains the command "C:\Users\anaranjo\portable soft\PortableGit>cd C:\Users\anaranjo\portable soft\mysql-9.5.0-win64\bin". The "Símbolo del sistema" window has a black background and contains the command "C:\Users\anaranjo\portable soft\mysql-9.5.0-win64\bin>.\mysqld". The taskbar also displays system icons for battery, signal, and date/time (25/01/2026, 12:49).

Se inicia el servidor de base de datos MySQL Server. Se opta por elegir un cliente en modo texto (Línea de comandos MySQL).

```
C:\Users\anaranjo\portable soft\mysql-9.5.0-win64\bin>.\mysql -u root -p
Enter password: **** 2
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 9.5.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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.

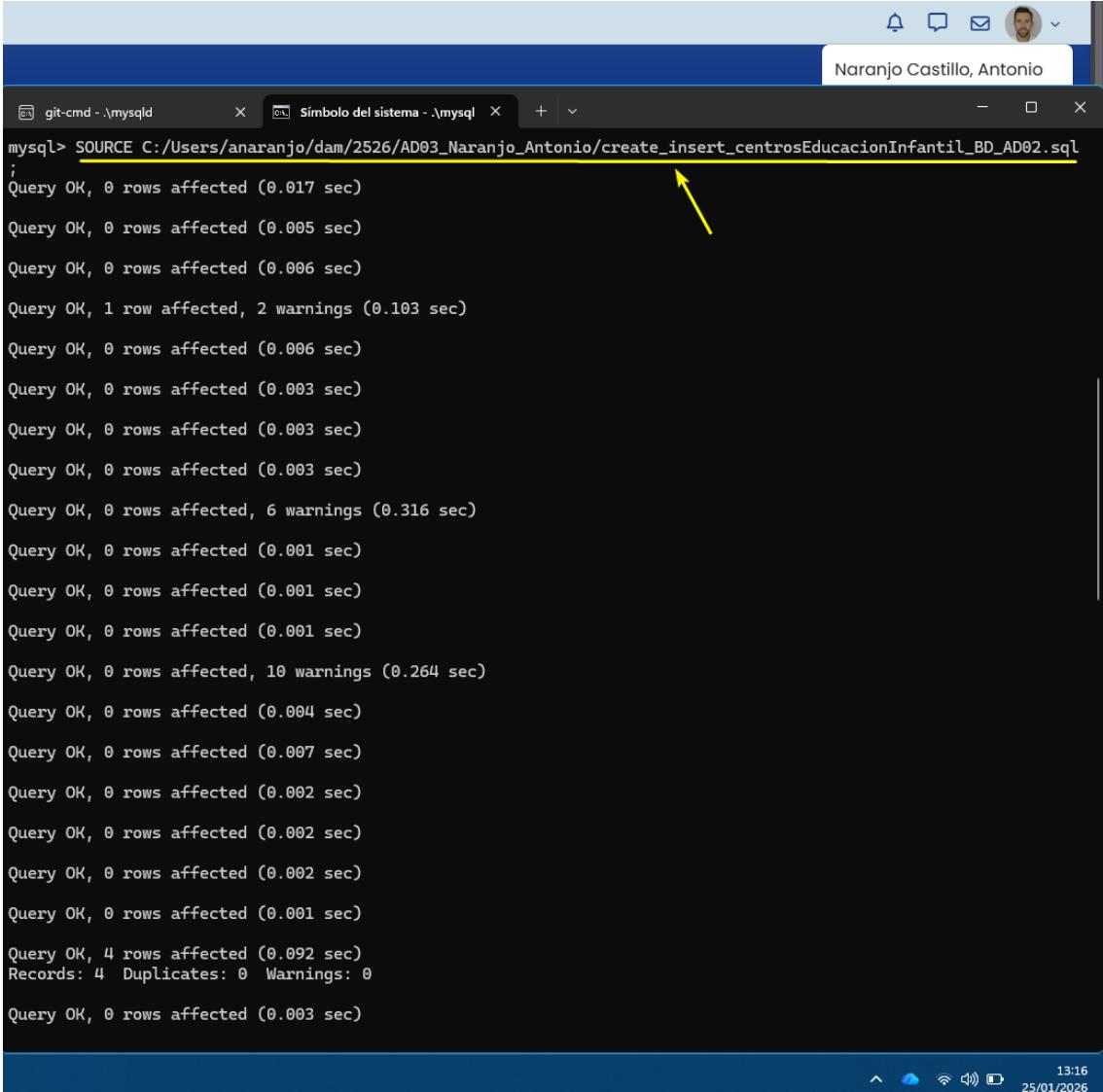
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES; 3
+-----+
| Database |
+-----+
| campeonato_atletismo |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.017 sec)

mysql> SHOW VARIABLES LIKE 'character_set%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | cp850 |
| character_set_connection | cp850 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | cp850 |
| character_set_server | utf8mb4 |
| character_set_system | utf8mb3 |
| character_sets_dir | C:\Users\anaranjo\portable soft\mysql-9.5.0-win64\share\charsets\ |
+-----+
8 rows in set, 1 warning (0.156 sec)

mysql> SET NAMES 'utf8mb4';
Query OK, 0 rows affected (0.019 sec)
```

En una segunda ventana del Símbolo del sistema (cmd) se accede a MySQL Server mediante usuario ‘root’ se introduce contraseña y se muestran las bases de datos actualmente existentes. De manera opcional, se establece la configuración UTF-8 para la correcta lectura en pantalla de caracteres especiales.



```
mysql> SOURCE C:/Users/anaranjo/dam/2526/AD03_Naranjo_Antonio/create_insert_centrosEducacionInfantil_BD_AD02.sql
;
Query OK, 0 rows affected (0.017 sec)

Query OK, 0 rows affected (0.005 sec)

Query OK, 0 rows affected (0.006 sec)

Query OK, 1 row affected, 2 warnings (0.103 sec)

Query OK, 0 rows affected (0.006 sec)

Query OK, 0 rows affected (0.003 sec)

Query OK, 0 rows affected (0.003 sec)

Query OK, 0 rows affected (0.003 sec)

Query OK, 0 rows affected, 6 warnings (0.316 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected, 10 warnings (0.264 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.007 sec)

Query OK, 0 rows affected (0.002 sec)

Query OK, 0 rows affected (0.002 sec)

Query OK, 0 rows affected (0.002 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 4 rows affected (0.092 sec)
Records: 4  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.003 sec)
```

Se lanza el script SQL de creación del esquema de la base de datos (Educacion_Infantil) según la sintaxis MySQL que se indica en la imagen.

Se muestran las bases de datos existentes nuevamente, para comprobar que el script se lanzó con éxito, se establece la nueva base de datos **centroseducacioninfantil**, se muestran las tablas que contiene y los tipos de datos de la primera **guardería**.

```

git-cmd - \mysqld      Símbolo del sistema - \mysql
4 rows in set (0.033 sec)
mysql> DESCRIBE educador_infantil;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| dni   | varchar(9) | NO  | PRI | NULL    |       |
| nombre | varchar(30) | YES |     | NULL    |       |
| apellidos | varchar(70) | YES |     | NULL    |       |
| fecha_nacimiento | date | YES |     | NULL    |       |
| fecha_alta | date | YES |     | NULL    |       |
| salario | float  | YES |     | NULL    |       |
| codigo_guarderia | varchar(5) | NO  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.030 sec)

mysql> SELECT * FROM guarderia;
+-----+-----+-----+
| codigo | nombre           | capacidad | presupuesto |
+-----+-----+-----+
| ABC01  | Centro Aventura | 50        | 20000       |
| ABC02  | Los arrecifes   | 70        | 45000       |
| CDE03  | Barcos de Papel | 30        | 15000       |
| DFG04  | La casita de Noa | 40        | 40000       |
+-----+-----+-----+
4 rows in set (0.007 sec)

mysql> SELECT * FROM educador_infantil;
+-----+-----+-----+-----+-----+-----+-----+
| dni   | nombre          | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+-----+-----+-----+-----+-----+-----+
| 10101010L | ELISA          | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000  | DFG04
| 11111111A | PEPE           | LOPEZ MARIN   | 1984-03-02 | 2010-03-01 | 35000  | ABC01
| 22222222B | LOLA           | PEREZ PEREZ   | 1988-01-01 | 2010-03-01 | 32000  | ABC01
| 33333333C | REMEDIOS        | COSTA RUBIO   | 1996-03-03 | 2015-01-01 | 25000  | ABC01
| 44444444D | LAURA          | GOMEZ GOMEZ   | 1978-01-20 | 2010-08-01 | 40000  | ABC02
| 55555555F | JUAN           | LOPEZ LOPEZ   | 1988-05-01 | 2008-05-03 | 25000  | ABC02
| 66666666G | ANA            | MARIN MARIN   | 2000-03-02 | 2020-09-01 | 18000  | ABC02
| 77777777H | CARMEN         | CASAS CAVA    | 1975-09-25 | 2000-05-08 | 35000  | CDE03
| 88888888J | MARCOS         | LUNA PIO      | 1989-08-15 | 2010-01-01 | 38000  | CDE03
| 99999999K | JUANA          | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500  | DFG04
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.099 sec)

mysql> |

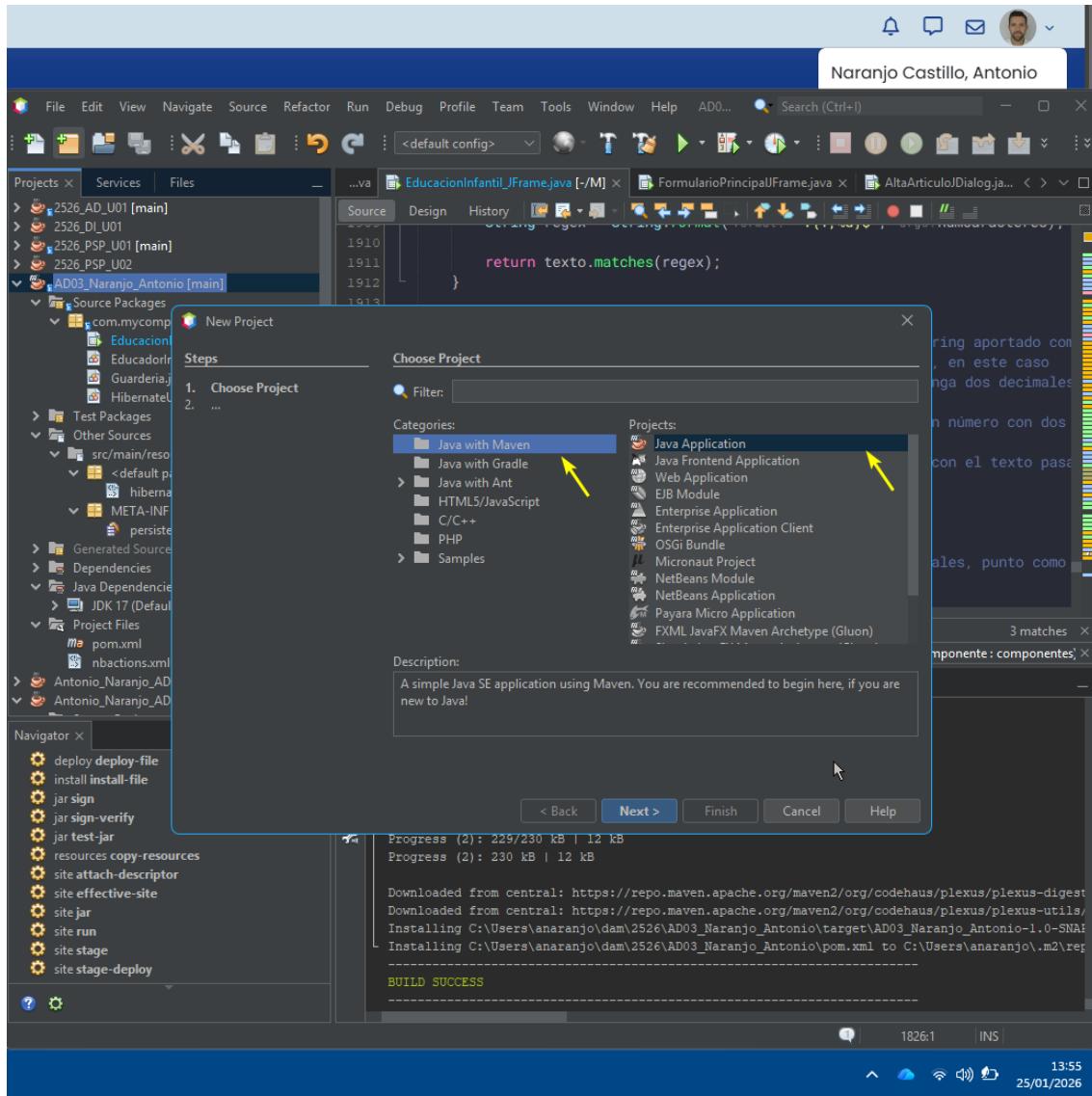
```

13:22
25/01/2026

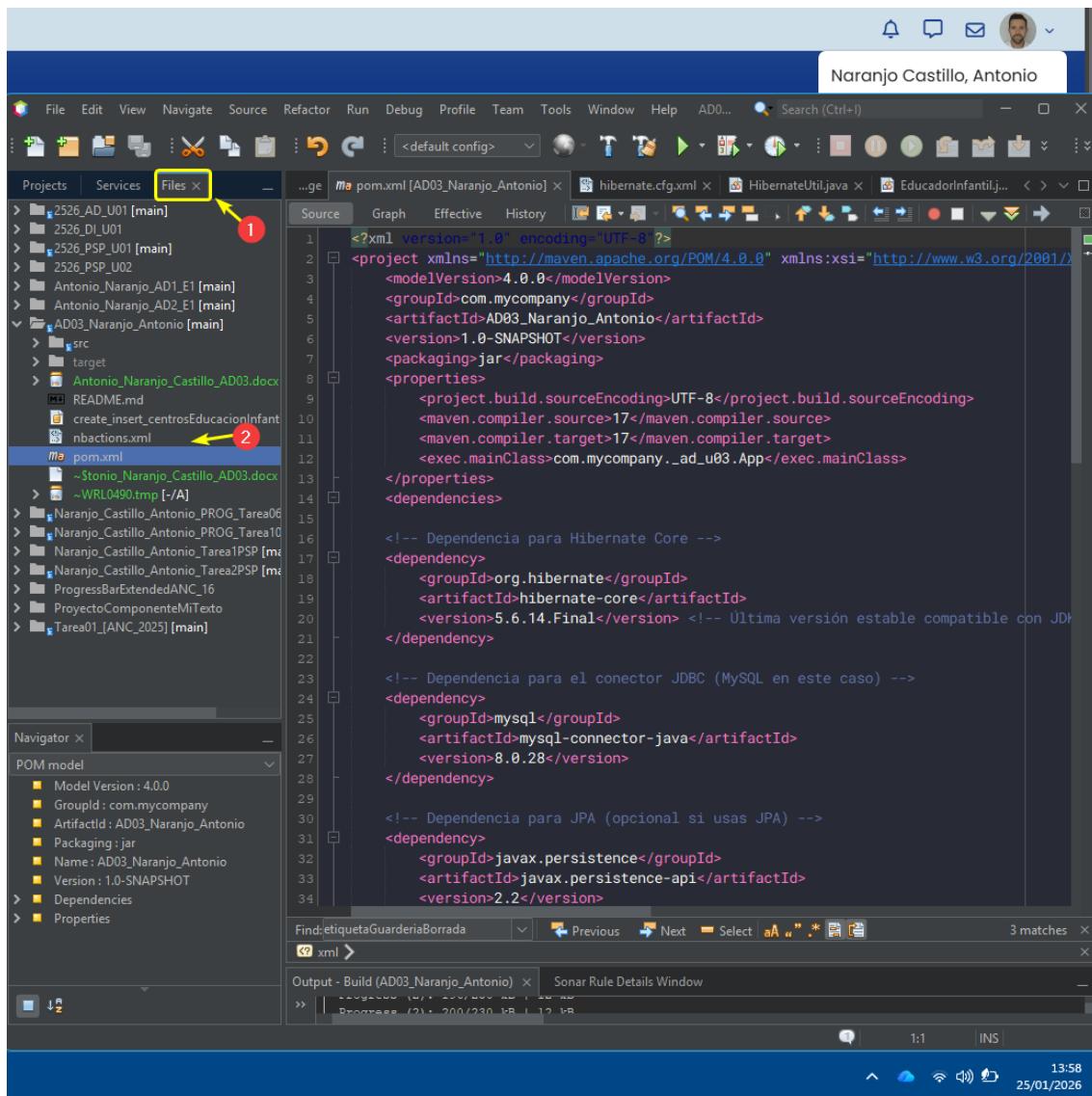
Se muestran los tipos de datos de la segunda tabla **educador_infantil**, y, se muestran los datos existentes de cada una de las tablas.

De esta manera, se puede asegurar que se han creado ambas tablas y se han insertado los registros correctamente.

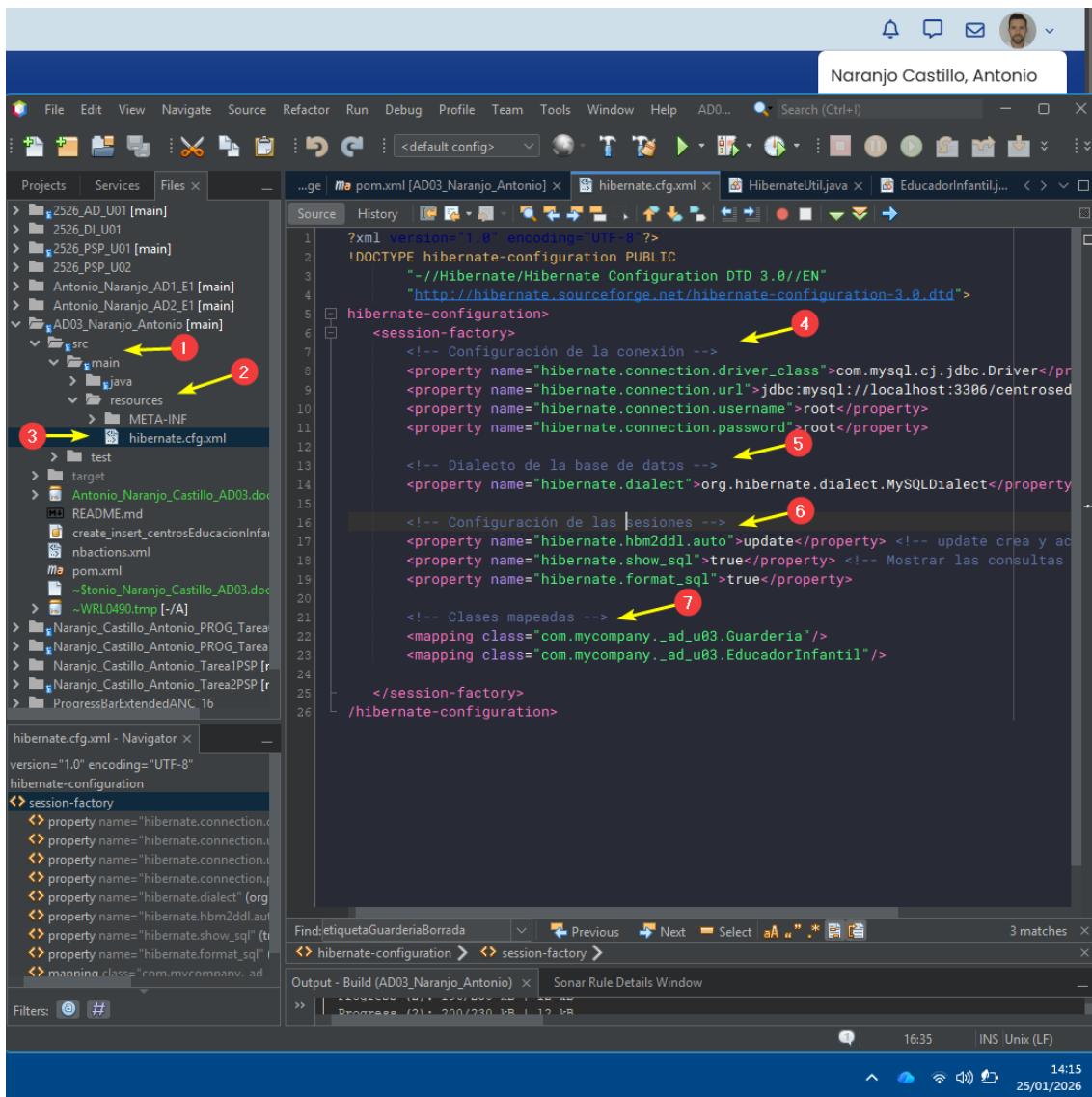
b) Desarrollar un proyecto en Netbeans con nombre *AD03_Apellido1_Nombre* y configurar Hibernate para poder realizar el mapeo de la base de datos creada en el apartado anterior. Debes detallar todo lo realizado referente al mapeo (creación del fichero de configuración *hibernate.cfg.xml*, generación del fichero de persistencia, etc.).



Se crea un proyecto Netbeans creando una aplicación **Java with Maven** para importar todas las librerías necesarias de manera automática.

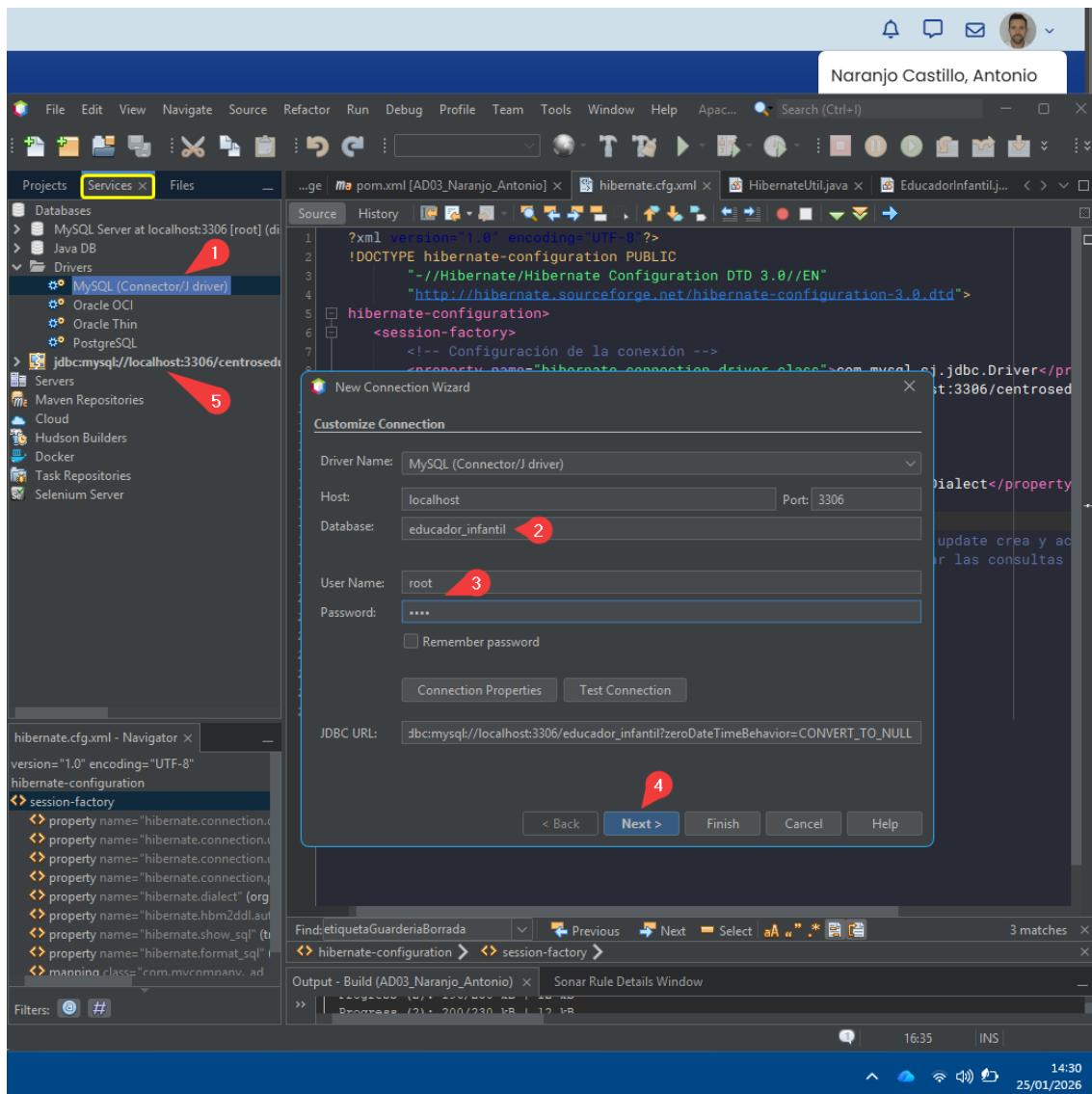


Se configuran las dependencias, para ello, en el explorador de archivos de Netbeans se accede al archivo **pom.xml** (Files→pom.xml) y se añaden las dependencias tales como: importación de hibernate, conector JDBC MySQL, persistencia y transacciones, según puede verse en la imagen aportada. Tras guardar el proyecto se muestran las dependencias necesarias cargándose las librerías automáticamente.

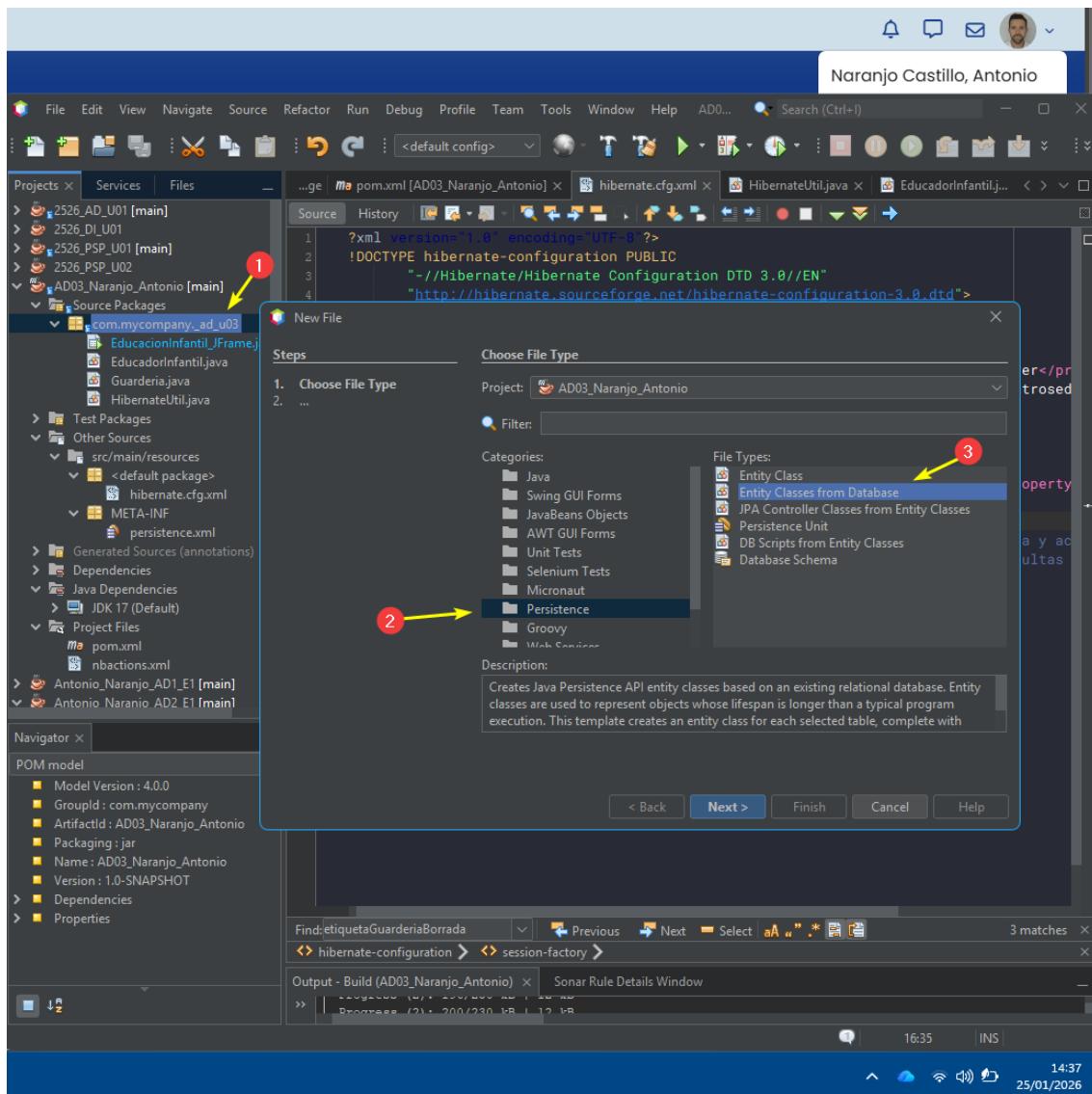


Se procede a definir el archivo de configuración de hibernate. Se crea la carpeta **resources** y dentro de ella un nuevo archivo XML bien formado denominado **hibernate.cfg.xml**. En este nuevo fichero se definen los datos de la conexión y las características de cómo va a trabajar hibernate, se trata de la herramienta que convierte los objetos en tablas y viceversa. Se definen: el driver MySQL, la ubicación de la base de datos **localhost** (en mi propia máquina) y puerto **3306**, el nombre de la base de datos **centroseducacioninfantil**, así como el nombre de usuario y contraseña con la cual se va a realizar la conexión. También se define el **dialecto** que empleará hibernate y la configuración de las sesiones, es decir, hibernate actualizará los datos en la base de datos tras llevar a cabo los cambios realizados en los atributos de los objetos de las clases mapeadas, y viceversa, los cambios surgidos en la base de datos se establecerán en los objetos. También se configura la manera como se muestran las sentencias que hibernate va ejecutando y formateadas para facilitar su legibilidad.

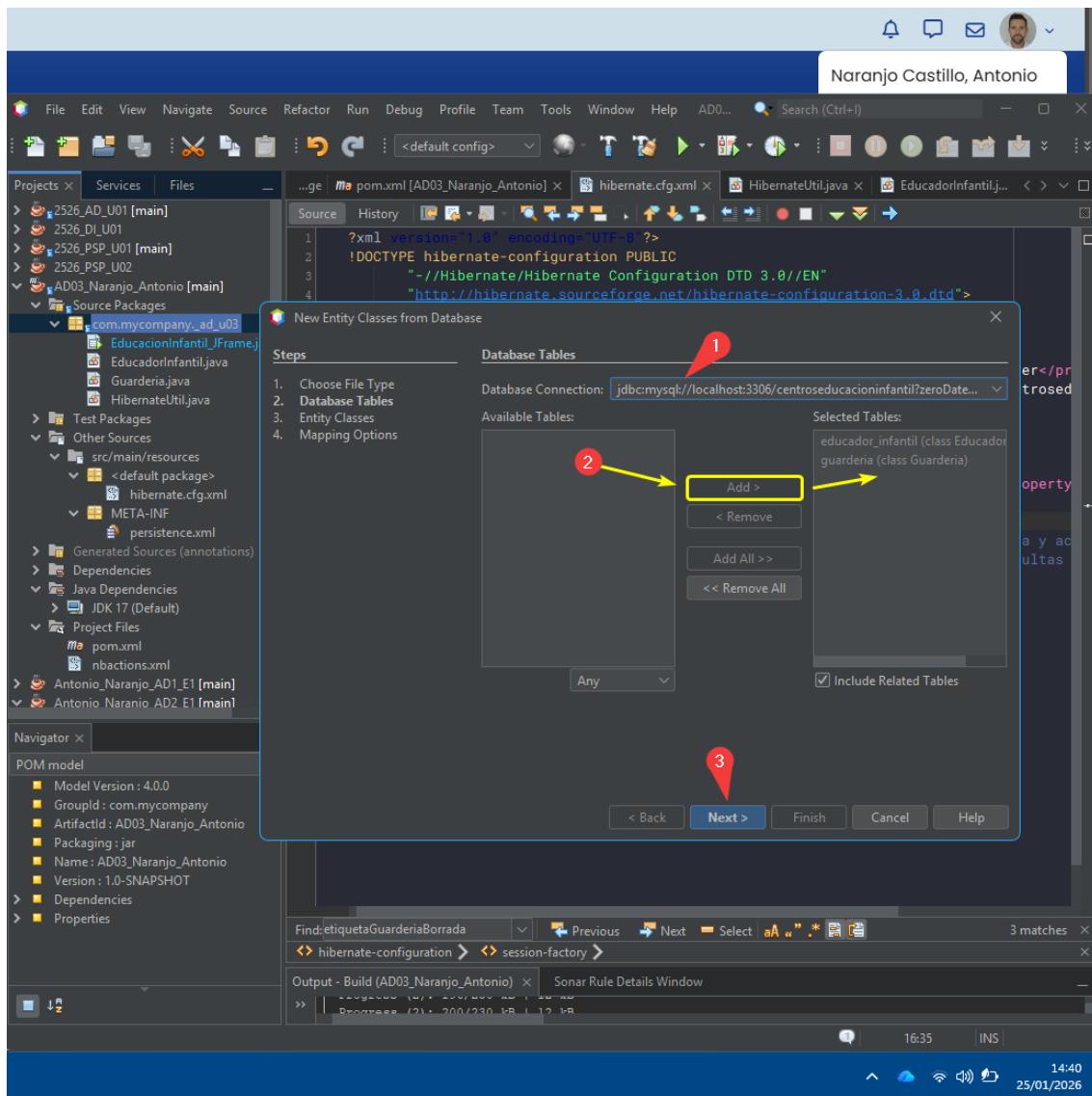
Y por último se definen las clases mapeadas que trabajarán sobre la base de datos.



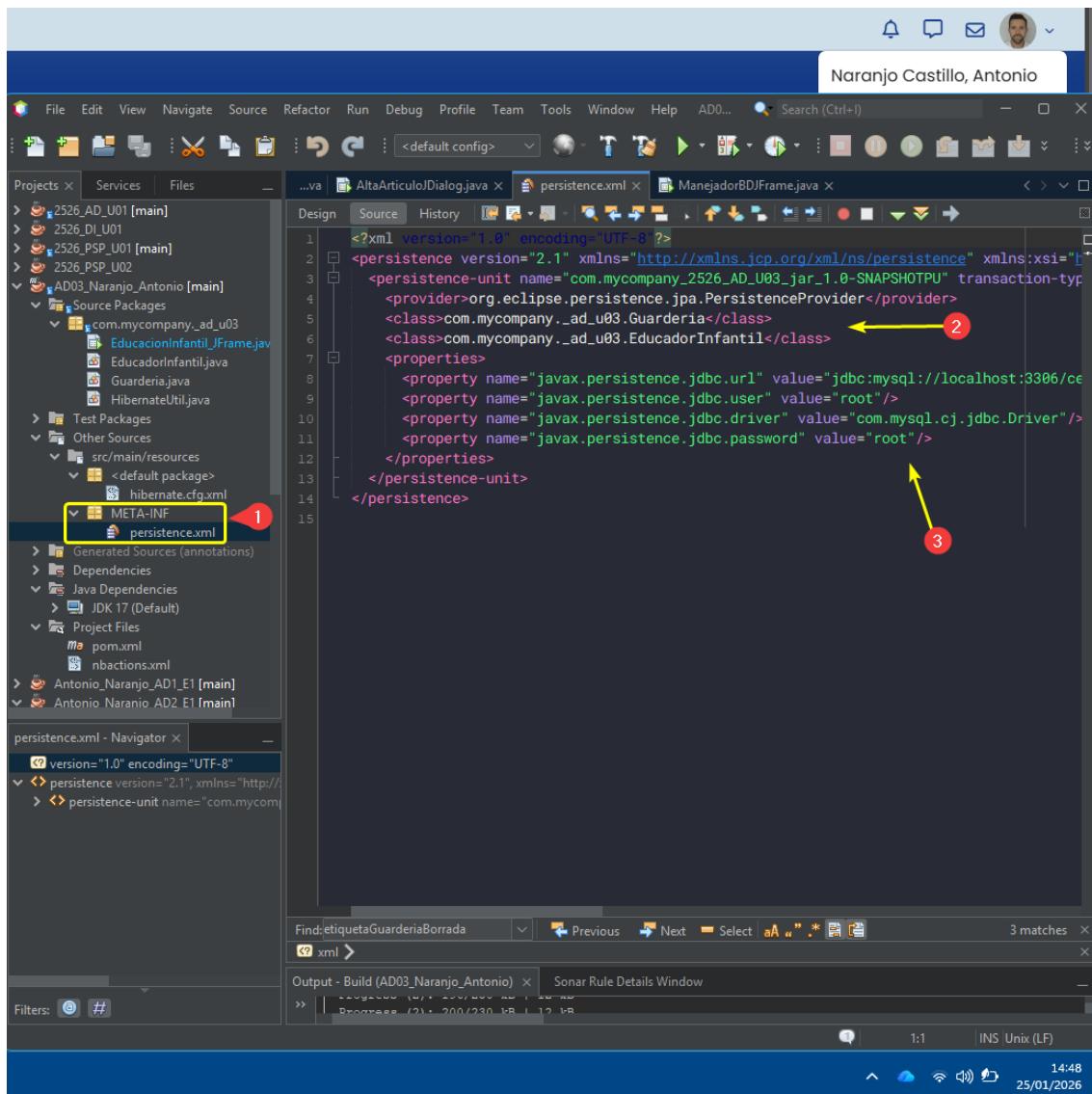
Se crea la conexión navegando por el explorador de archivos de Netbeans Services → Databases → Drivers → Connetor/J driver y pulsado botón derecho del ratón se cliquea sobre Connect Using... En la ventana emergente resultante (según se muestra en la imagen) se definen la base de datos y los datos de usuario que realiza la conexión, posteriormente se pulsa sobre next hasta terminar el proceso, el resultado será la conexión resultante.



Se crea el archivo de persistencia para que los datos y los cambios realizados queden registrados con éxito y perduren tras la finalización del aplicativo. Para ello, a través del navegador de Netbeans en el apartado **Projects** se cliquea botón derecho sobre el paquete que contiene la aplicación java en cuestión y en ventana emergente resultante se accede por medio de **Other** a la categoría **Persistence** y al tipo de archivo **Entity Classes from Databases** y se pulsa en **Next**.



Luego se selecciona la conexión a la base de datos en Available Tables se muestran las clases disponible y se añaden a Selected Tables para incorporarlas.



Por último, se chequean las propiedades deseadas para la generación del archivo de persistencia en el siguiente paso de la ventana emergente anterior y pulsando en Next se creará el archivo de persistencia.

Destacar que, se muestran las clases mapeadas anteriormente seleccionadas y que serán las mismas que aparecen en el **archivo de configuración de hibernate**. En el explorador de archivos de Netbeans se pueden visualizar las clases mapeadas **EducadorInfantil** y **Guarderia** generadas junto al archivo de persistencia.

```

10 /**
11  * Clase para la fabricación de sesiones de Hibernate. Implementa una variable
12  * sessionFactory de manera static para asegurar que solo exista una instancia
13  * de {code SessionFactory} en toda la aplicación, optimizando así el uso de
14  * recursos y la conexión con el SGBD MySQL.
15  *
16  * @author Antonio Naranjo Castillo
17  * @version 1.0
18  */
19 public class HibernateUtil {
20
21     // Declaración de la variable fábrica de sesiones
22     private static SessionFactory sessionFactory;
23
24     /**
25      * Bloqueo estático que inicia la configuración de hibernate.
26      * Lee el archivo de configuración y construye la fábrica de sesiones.
27      */
28     static {
29         try {
30             sessionFactory = new Configuration().configure().buildSessionFactory();
31         } catch (HibernateException ex) {
32             System.out.println(ex.getMessage());
33         }
34     }
35
36     // Método getter que devuelve la fábrica de sesiones.
37     public static SessionFactory getSessionFactory() {
38         return sessionFactory;
39     }
40 }
41
42
43

```

Se crea la clase `HibernateUtil` que permite crear el objeto `SessionFactory` (fabricante de sesiones) con el cual se pueden generar objetos `Session` que permiten interaccionar con la base de datos. Estos objetos `Session` permiten establecer la comunicación con la base de datos dando acceso a los datos bidireccionalmente. Esta clase `HibernateUtil` se ubica dentro del paquete donde se encuentran las clases mapeadas anteriores según se muestran en la imagen.

EJERCICIO 2:

Desarrolla opciones en el proyecto creado en el ejercicio anterior que den la posibilidad de realizar las siguientes acciones (se debe hacer con una interfaz gráfica usando swing):

a) Añadir nuevas guarderías y nuevos educadores infantiles a la base de datos.

1) Añadir Objetos

Guarderías

- Código: HJK05
- Nombre: EL CASTILLO DE MI BEBÉ
- Capacidad: 50
- Presupuesto: 35000.0

Educadores

- DNI:
- Nombre:
- Apellidos:
- Código Guardería:

Guardería añadida con éxito.

2) Eliminar Objetos

Guarderías

- Código: ABC01
- Borrar
- Nombre de la guardería a eliminar:

Educadores

- DNI: 11111111A
- Borrar
- Nombre del educador a eliminar:

3) Listar Guarderías

Listado de guarderías

4) Listar Educadores

Centro Aventura

Listado de educadores

5) Listar Educadores

Salario mayor que:

Listado de educadores

6) Actualizar salario Educador Infantil

DNI: 11111111A

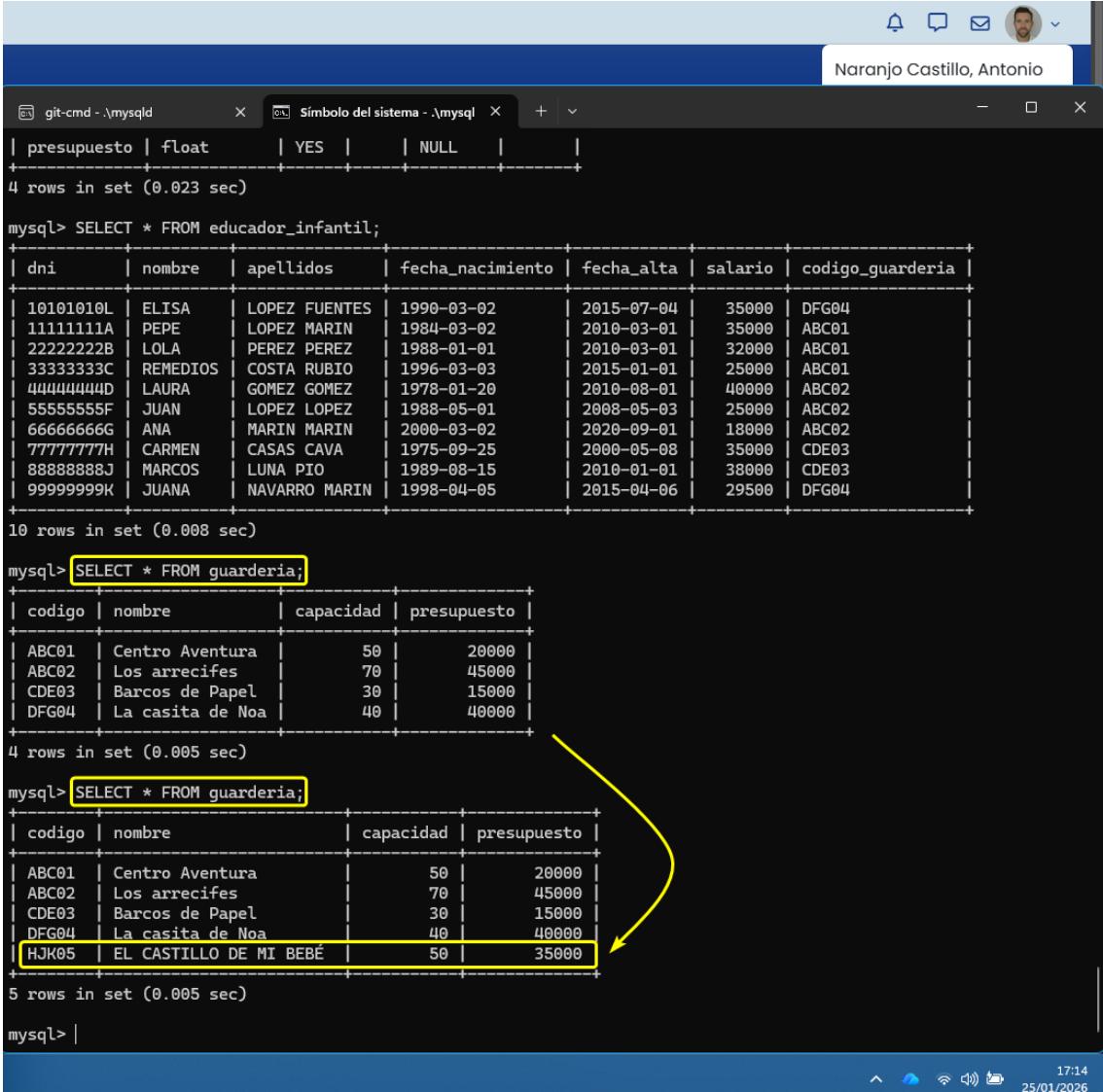
Nuevo Salario:

Nombre del educador a eliminar:

17:07 25/01/2026

Se elabora una interfaz gráfica sencilla a modo de pizarra mostrando todas las actividades que se llevarán a cabo en la tarea.

En la indicación de la imagen se procede a pulsar sobre el botón Añadir, para agregar a la base de datos una nueva guardería según los atributos que se muestran en la imagen.



```

git-cmd - \mysql   Símbolo del sistema - \mysql
+-----+-----+-----+-----+
| presupuesto | float | YES | NULL |
+-----+-----+-----+-----+
4 rows in set (0.023 sec)

mysql> SELECT * FROM educador_infantil;
+-----+-----+-----+-----+-----+-----+-----+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 40000 | ABC02
| 55555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03
| 99999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04
+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.008 sec)

mysql> SELECT * FROM guarderia;
+-----+-----+-----+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+-----+-----+-----+
| ABC01 | Centro Aventura | 50 | 20000 |
| ABC02 | Los arrecifes | 70 | 45000 |
| CDE03 | Barcos de Papel | 30 | 15000 |
| DFG04 | La casita de Noa | 40 | 40000 |
+-----+-----+-----+-----+
4 rows in set (0.005 sec)

mysql> SELECT * FROM guarderia;
+-----+-----+-----+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+-----+-----+-----+
| ABC01 | Centro Aventura | 50 | 20000 |
| ABC02 | Los arrecifes | 70 | 45000 |
| CDE03 | Barcos de Papel | 30 | 15000 |
| DFG04 | La casita de Noa | 40 | 40000 |
| HJK05 | EL CASTILLO DE MI BEBÉ | 50 | 35000 |
+-----+-----+-----+-----+
5 rows in set (0.005 sec)

mysql> |

```

Se puede comprobar que los datos fueron actualizados en la base de datos apareciendo la nueva guardería creada. Se realiza una consulta previa y otra posterior a la ejecución del botón Añadir del apartado Guarderías de la interfaz Swing.

```

1058         title: "Solicitud del presupuesto",
1059         messageType: JOptionPane.QUESTION_MESSAGE
1060     );
1061 }
1062 ppgTextField.setText( t:presuGuarderia);
1063
1064 }
1065
1066 // Se inicia una sesión
1067 try {
1068     // Solicita a la factoría una nueva sesión de trabajo con la base de datos
1069     Session session = HibernateUtil.getSessionFactory().openSession();
1070     // Se llama el método agregarGuarderia() para añadir una guardería a la base
1071     agregarGuarderia( ss:session, codigo, nombre, capacidad, presupuesto);
1072     // Informa visualmente al usuario en la interfaz
1073     jLabelAgregarGuarderia.setText( text:"Guardería añadida con éxito.");
1074     // Se cierra la sesión
1075     session.close();
1076
1077     // Captura cualquier excepción específica de Hibernate
1078 } catch (HibernateException e) {
1079     System.out.println( x:e.getMessage());
1080
1081     // Se refrescan los combobox y se selecciona el primer ítem por defecto
1082     refrescarComboBoxGuarderiasUnicas();
1083     refrescarComboBoxNombreGuarderias();
1084     jComboBoxGuarderias.setSelectedIndex( anIndex:0);
1085 }
1086
1087 /**
1088 * Restablece los campos de entrada del panel de añadir educadores.
1089 *
1090 * @param evt Evento de acción.
1091 */

```

Find:etiquetaGuarderiaBorrada Previous Next Select aA " " Find... com.mycompany_ad_u03.EducacionInfantil.JFrame > agregarGuarderiaButtonActionPerformed > nombre >

Output - Run (AD03_Naranjo_Antonio) Sonar Rule Details Window

Tras pulsar en Añadir en el apartado Guarderías, el código que se lleva a cabo es el siguiente: Posterior a la correcta validación de los datos de entrada, el método crea un objeto Session obtenido del objeto SessionFactory (fábrica de sesiones) de la clase HibernateUtil y posteriormente se ejecuta el método agregarGuarderia() aportando como argumentos los atributos del objeto Guarderia que se pretende añadir. Luego una vez agregados los nuevos datos se procede a refrescar los objetos ComboBox que existan en la interfaz Swing y que estén afectados por tales cambios.

The screenshot shows the NetBeans IDE interface with the following details:

- Projects View:** Shows multiple projects including "2526_AD_U01 [main]", "2526_DL_U01", "2526_PSP_U01 [main]", "2526_PSP_U02", "AD03_Naranjo_Antonio [main]" (selected), and "Antonio_Naranjo_E1 [main]".
- Source Editor:** Displays the `EducacionInfantilJFrame.java` file with the following code snippet:

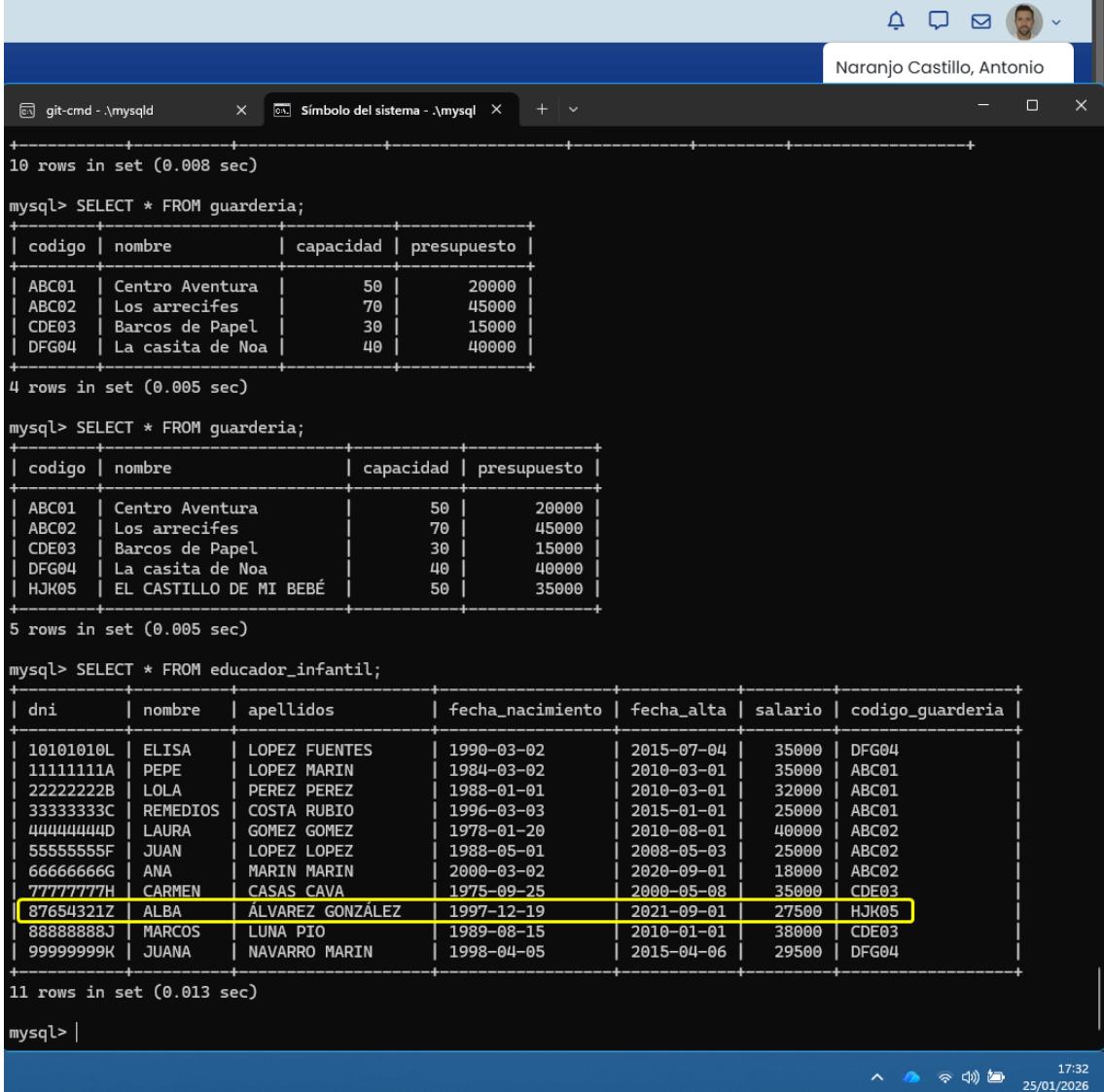

```

1307 /**
1308 * static void agregarGuarderia(Session ss, String codigo, String nombre, Integer capacidad)
1309 *
1310 // Inicia una unidad de trabajo lógica para asegurar la integridad de los datos
1311 Transaction tst = ss.beginTransaction();
1312
1313 try {
1314
1315     // Instancia el objeto 'Guarderia' usando el código como identificador (PK)
1316     // Se establecen los atributos de una entidad Guarderia
1317     Guarderia guarde = new Guarderia(codigo);
1318     guarde.setNombre(nombre);
1319     guarde.setCapacidad(capacidad);
1320     guarde.setPresupuesto(presupuesto);
1321     // Pasa el objeto al estado persistente para que Hibernate lo gestione
1322     ss.save(garde);
1323
1324     /// Confirma los cambios de forma definitiva en la base de datos MySQL
1325     tst.commit();
1326
1327 } catch (ConstraintViolationException e) {
1328
1329     if (tst != null) {
1330         // Hacer rollback se revierte cualquier cambio pendiente para evitar daños
1331         tst.rollback();
1332     }
1333     System.out.println(e.getMessage());
1334
1335 } catch (Exception e) {
1336
1337     if (tst != null) {
1338         // Hacer rollback para cualquier otro error
1339         tst.rollback();
1340     }
1341 }
      
```
- Navigator View:** Shows the members of the `agregarGuarderia` method.
- Output View:** Shows the output for the run configuration "Run (AD03_Naranjo_Antonio)".

En el método `agregarGuarderia()` que recibe como argumento todos los atributos del objeto `Guarderia`, es donde se tiene lugar la transacción. Se crea un objeto `Transaction`, luego se crea un objeto `guarderia`, se establecen sus atributos y se ejecuta el método `save()` para pasar los datos del objeto `Session` al estado persistente, luego, se ejecuta el método `commit()` del objeto `Transaction` para confirmar los cambios y así verlos reflejados en la base de datos, tal y como se ha comprobado anteriormente.

The screenshot shows a Windows application window titled "Educadores". The main form contains fields for DNI (87654321Z), Nombre (ALBA), Apellidos (ÁLVAREZ GONZÁLEZ), Fecha Nacimiento (19/12/1997), Fecha Alta (01/09/2021), and Salario (27500.00). A yellow arrow points to the "Añadir" button. Below the form, a message says "Educador infantil añadido con éxito.". On the left side of the window, there are other tabs like "Bebé" and "Listar". On the right side, there are sections for "4) Listar Educadores" and "6) Actualizar salario Educador Infantil". The status bar at the bottom shows the date and time: 25/01/2026, 17:30.

De la misma manera se procede para agregar un nuevo educador infantil.



```
git-cmd - \mysqld
Símbolo del sistema - \mysql
Naranjo Castillo, Antonio

mysql> SELECT * FROM guarderia;
+-----+-----+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+-----+-----+
| ABC01 | Centro Aventura | 50 | 20000 |
| ABC02 | Los arrecifes | 70 | 45000 |
| CDE03 | Barcos de Papel | 30 | 15000 |
| DFG04 | La casita de Noa | 40 | 40000 |
+-----+-----+-----+
4 rows in set (0.008 sec)

mysql> SELECT * FROM guarderia;
+-----+-----+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+-----+-----+
| ABC01 | Centro Aventura | 50 | 20000 |
| ABC02 | Los arrecifes | 70 | 45000 |
| CDE03 | Barcos de Papel | 30 | 15000 |
| DFG04 | La casita de Noa | 40 | 40000 |
| HJK05 | EL CASTILLO DE MI BEBÉ | 50 | 35000 |
+-----+-----+-----+
5 rows in set (0.005 sec)

mysql> SELECT * FROM educador_infantil;
+-----+-----+-----+-----+-----+-----+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+-----+-----+-----+-----+-----+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04 |
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01 |
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01 |
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01 |
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 40000 | ABC02 |
| 55555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02 |
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02 |
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03 |
| 87654321Z | ALBA | ÁLVAREZ GONZÁLEZ | 1997-12-19 | 2021-09-01 | 27500 | HJK05 |
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03 |
| 99999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04 |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.013 sec)

mysql> |
```

Se puede observar el nuevo educador infantil añadido a la base de datos.

```

921     }
922
923     // Se inicia un bloque de control de errores para manejar fallos en la base
924     try {
925         // Solicita a la factoría una nueva sesión de trabajo con la base de datos
926         Session session = HibernateUtil.getSessionFactory().openSession();
927         // Se llama el método agregarGuarderia() para añadir una guardería a la base de datos
928         // Ejecuta la lógica de guardado pasando los datos del formulario y la sesión
929         agregarEducador(ss, dni, nombre, apellidos, fechaNacimiento, fechaIngreso);
930         // Informa visualmente al usuario en la interfaz que la operación fue exitosa
931         jLabelAgregarEducador.setText("Educador infantil añadido con éxito");
932         // Libera los recursos y finaliza la conexión actual de Hibernate
933         session.close();
934
935         // Captura cualquier excepción específica de Hibernate que haya ocurrido en la ejecución
936         } catch (HibernateException e) {
937             System.out.println(e.getMessage());
938         }
939         // Se refrescan los combobox con la documentación actualizada apuntando al método
940         // refreshComboBoxEducadoresUnicos();
941         jComboBoxEducadores.setSelectedIndex(0);
942     }
943
944     /**
945      * Gestiona la eliminación de una guardería seleccionada. Obtiene el código
946      * del ComboBox de eliminación y ejecuta el borrado mediante Hibernate.
947      *
948      * @param evt Evento de acción.
949      */
950     private void eliminarGuarderiaButtonActionPerformed(java.awt.event.ActionEvent evt) {
951         // TODO add your handling code here
952
953         // Declaración de campos de la tabla Guarderia
954
955     }

```

The screenshot shows the NetBeans IDE interface with the code editor open to 'EducacionInfantil_JFrame.java'. The code implements logic for adding a new Educator to a database using Hibernate. It includes a try-catch block for Hibernate exceptions, session creation, and data insertion. The code also refreshes a dropdown menu after addition.

Lo más importante del código implementado tras accionar el botón Añadir del apartado Educadores es la creación del objeto Session, y la ejecución del método agregarEducador() alimentado como argumentos los atributos del objeto EducadorInfantil. Todo ello tras la previa validación de los datos introducidos por el usuario.

```

1355     /* @param configurarDni el objeto que pertenece */
1356     static void agregarEducador(Session ss, String dni, String nombre, String apellido) {
1357         // Se declara/instancia un objeto tipo Transaction
1358         Transaction tst = ss.beginTransaction();
1359
1360         try {
1361             // Instancia el objeto 'EducadorInfantil' usando el código como identificador
1362             EducadorInfantil educador = new EducadorInfantil(dni);
1363             educador.setNombre(nombre);
1364             educador.setApellidos(apellido);
1365             educador.setFechaNacimiento(fechaNacimiento);
1366             educador.setFechaAlta(fechaAlta);
1367             educador.setSalario(salario);
1368             educador.setCodigoGuarderia(codigoGuarderia);
1369
1370             // Pasa el objeto al estado persistente
1371             ss.save(educador);
1372
1373             // Confirma los cambios de forma definitiva
1374             tst.commit();
1375
1376         } catch (ConstraintViolationException e) {
1377
1378             if (tst != null) {
1379                 // Hacer rollback si hay un error
1380                 tst.rollback();
1381             }
1382             System.out.println(e.getMessage());
1383
1384         } catch (Exception e) {
1385
1386             if (tst != null) {
1387                 // Hacer rollback para cualquier otro error
1388                 tst.rollback();
1389             }
1390         }
1391     }
1392
1393     /**
1394      * @param etiquetaGuarderiaBorrada
1395      */
1396     public void actualizarSalario(Session ss, String dni, String apellido, String nombre, String fechaNacimiento, String fechaAlta, Double salario, String codigoGuarderia) {
1397         EducadorInfantil educador = (EducadorInfantil) ss.get(EducadorInfantil.class, dni);
1398
1399         if (educador != null) {
1400             educador.setApellido(apellido);
1401             educador.setNombre(nombre);
1402             educador.setFechaNacimiento(fechaNacimiento);
1403             educador.setFechaAlta(fechaAlta);
1404             educador.setSalario(salario);
1405             educador.setCodigoGuarderia(codigoGuarderia);
1406
1407             ss.update(educador);
1408
1409             tst.commit();
1410         }
1411     }
1412
1413     /**
1414      * @param etiquetaGuarderiaBorrada
1415      */
1416     public void actualizarSueldoEducativo(Session ss, String dni, String apellido, String nombre, String fechaNacimiento, String fechaAlta, Double salario, String codigoGuarderia) {
1417         EducadorInfantil educador = (EducadorInfantil) ss.get(EducadorInfantil.class, dni);
1418
1419         if (educador != null) {
1420             educador.setApellido(apellido);
1421             educador.setNombre(nombre);
1422             educador.setFechaNacimiento(fechaNacimiento);
1423             educador.setFechaAlta(fechaAlta);
1424             educador.setSalario(salario);
1425             educador.setCodigoGuarderia(codigoGuarderia);
1426
1427             ss.update(educador);
1428
1429             tst.commit();
1430         }
1431     }
1432
1433     /**
1434      * @param etiquetaGuarderiaBorrada
1435      */
1436     public void agregarEducador(Session ss, String dni, String apellido, String nombre, String fechaNacimiento, String fechaAlta, Double salario, String codigoGuarderia) {
1437         EducadorInfantil educador = new EducadorInfantil(dni);
1438
1439         if (educador != null) {
1440             educador.setApellido(apellido);
1441             educador.setNombre(nombre);
1442             educador.setFechaNacimiento(fechaNacimiento);
1443             educador.setFechaAlta(fechaAlta);
1444             educador.setSalario(salario);
1445             educador.setCodigoGuarderia(codigoGuarderia);
1446
1447             ss.save(educador);
1448
1449             tst.commit();
1450         }
1451     }
1452
1453     /**
1454      * @param etiquetaGuarderiaBorrada
1455      */
1456     public void borrarEducador(Session ss, String dni) {
1457         EducadorInfantil educador = (EducadorInfantil) ss.get(EducadorInfantil.class, dni);
1458
1459         if (educador != null) {
1460             ss.delete(educador);
1461
1462             tst.commit();
1463         }
1464     }
1465
1466     /**
1467      * @param etiquetaGuarderiaBorrada
1468      */
1469     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1470         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1471
1472         if (guarderia != null) {
1473             ss.delete(guarderia);
1474
1475             tst.commit();
1476         }
1477     }
1478
1479     /**
1480      * @param etiquetaGuarderiaBorrada
1481      */
1482     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1483         Guarderia guarderia = new Guarderia(codigoGuarderia);
1484
1485         if (guarderia != null) {
1486             ss.save(guarderia);
1487
1488             tst.commit();
1489         }
1490     }
1491
1492     /**
1493      * @param etiquetaGuarderiaBorrada
1494      */
1495     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1496         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1497
1498         if (guarderia != null) {
1499             guarderia.setNombre(nombre);
1500
1501             ss.update(guarderia);
1502
1503             tst.commit();
1504         }
1505     }
1506
1507     /**
1508      * @param etiquetaGuarderiaBorrada
1509      */
1510     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1511         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1512
1513         if (guarderia != null) {
1514             ss.delete(guarderia);
1515
1516             tst.commit();
1517         }
1518     }
1519
1520     /**
1521      * @param etiquetaGuarderiaBorrada
1522      */
1523     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1524         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1525
1526         if (guarderia != null) {
1527             guarderia.setSalario(salario);
1528
1529             ss.update(guarderia);
1530
1531             tst.commit();
1532         }
1533     }
1534
1535     /**
1536      * @param etiquetaGuarderiaBorrada
1537      */
1538     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1539         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1540
1541         if (guarderia != null) {
1542             ss.delete(guarderia);
1543
1544             tst.commit();
1545         }
1546     }
1547
1548     /**
1549      * @param etiquetaGuarderiaBorrada
1550      */
1551     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1552         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1553
1554         if (guarderia != null) {
1555             guarderia.setSalario(salario);
1556
1557             ss.update(guarderia);
1558
1559             tst.commit();
1560         }
1561     }
1562
1563     /**
1564      * @param etiquetaGuarderiaBorrada
1565      */
1566     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1567         Guarderia guarderia = new Guarderia(codigoGuarderia);
1568
1569         if (guarderia != null) {
1570             ss.save(guarderia);
1571
1572             tst.commit();
1573         }
1574     }
1575
1576     /**
1577      * @param etiquetaGuarderiaBorrada
1578      */
1579     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1580         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1581
1582         if (guarderia != null) {
1583             guarderia.setNombre(nombre);
1584
1585             ss.update(guarderia);
1586
1587             tst.commit();
1588         }
1589     }
1590
1591     /**
1592      * @param etiquetaGuarderiaBorrada
1593      */
1594     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1595         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1596
1597         if (guarderia != null) {
1598             ss.delete(guarderia);
1599
1600             tst.commit();
1601         }
1602     }
1603
1604     /**
1605      * @param etiquetaGuarderiaBorrada
1606      */
1607     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1608         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1609
1610         if (guarderia != null) {
1611             guarderia.setSalario(salario);
1612
1613             ss.update(guarderia);
1614
1615             tst.commit();
1616         }
1617     }
1618
1619     /**
1620      * @param etiquetaGuarderiaBorrada
1621      */
1622     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1623         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1624
1625         if (guarderia != null) {
1626             ss.delete(guarderia);
1627
1628             tst.commit();
1629         }
1630     }
1631
1632     /**
1633      * @param etiquetaGuarderiaBorrada
1634      */
1635     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1636         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1637
1638         if (guarderia != null) {
1639             guarderia.setSalario(salario);
1640
1641             ss.update(guarderia);
1642
1643             tst.commit();
1644         }
1645     }
1646
1647     /**
1648      * @param etiquetaGuarderiaBorrada
1649      */
1650     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1651         Guarderia guarderia = new Guarderia(codigoGuarderia);
1652
1653         if (guarderia != null) {
1654             ss.save(guarderia);
1655
1656             tst.commit();
1657         }
1658     }
1659
1660     /**
1661      * @param etiquetaGuarderiaBorrada
1662      */
1663     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1664         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1665
1666         if (guarderia != null) {
1667             guarderia.setNombre(nombre);
1668
1669             ss.update(guarderia);
1670
1671             tst.commit();
1672         }
1673     }
1674
1675     /**
1676      * @param etiquetaGuarderiaBorrada
1677      */
1678     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1679         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1680
1681         if (guarderia != null) {
1682             ss.delete(guarderia);
1683
1684             tst.commit();
1685         }
1686     }
1687
1688     /**
1689      * @param etiquetaGuarderiaBorrada
1690      */
1691     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1692         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1693
1694         if (guarderia != null) {
1695             guarderia.setSalario(salario);
1696
1697             ss.update(guarderia);
1698
1699             tst.commit();
1700         }
1701     }
1702
1703     /**
1704      * @param etiquetaGuarderiaBorrada
1705      */
1706     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1707         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1708
1709         if (guarderia != null) {
1710             ss.delete(guarderia);
1711
1712             tst.commit();
1713         }
1714     }
1715
1716     /**
1717      * @param etiquetaGuarderiaBorrada
1718      */
1719     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1720         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1721
1722         if (guarderia != null) {
1723             guarderia.setSalario(salario);
1724
1725             ss.update(guarderia);
1726
1727             tst.commit();
1728         }
1729     }
1730
1731     /**
1732      * @param etiquetaGuarderiaBorrada
1733      */
1734     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1735         Guarderia guarderia = new Guarderia(codigoGuarderia);
1736
1737         if (guarderia != null) {
1738             ss.save(guarderia);
1739
1740             tst.commit();
1741         }
1742     }
1743
1744     /**
1745      * @param etiquetaGuarderiaBorrada
1746      */
1747     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1748         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1749
1750         if (guarderia != null) {
1751             guarderia.setNombre(nombre);
1752
1753             ss.update(guarderia);
1754
1755             tst.commit();
1756         }
1757     }
1758
1759     /**
1760      * @param etiquetaGuarderiaBorrada
1761      */
1762     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1763         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1764
1765         if (guarderia != null) {
1766             ss.delete(guarderia);
1767
1768             tst.commit();
1769         }
1770     }
1771
1772     /**
1773      * @param etiquetaGuarderiaBorrada
1774      */
1775     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1776         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1777
1778         if (guarderia != null) {
1779             guarderia.setSalario(salario);
1780
1781             ss.update(guarderia);
1782
1783             tst.commit();
1784         }
1785     }
1786
1787     /**
1788      * @param etiquetaGuarderiaBorrada
1789      */
1790     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1791         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1792
1793         if (guarderia != null) {
1794             ss.delete(guarderia);
1795
1796             tst.commit();
1797         }
1798     }
1799
1800     /**
1801      * @param etiquetaGuarderiaBorrada
1802      */
1803     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1804         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1805
1806         if (guarderia != null) {
1807             guarderia.setSalario(salario);
1808
1809             ss.update(guarderia);
1810
1811             tst.commit();
1812         }
1813     }
1814
1815     /**
1816      * @param etiquetaGuarderiaBorrada
1817      */
1818     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1819         Guarderia guarderia = new Guarderia(codigoGuarderia);
1820
1821         if (guarderia != null) {
1822             ss.save(guarderia);
1823
1824             tst.commit();
1825         }
1826     }
1827
1828     /**
1829      * @param etiquetaGuarderiaBorrada
1830      */
1831     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1832         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1833
1834         if (guarderia != null) {
1835             guarderia.setNombre(nombre);
1836
1837             ss.update(guarderia);
1838
1839             tst.commit();
1840         }
1841     }
1842
1843     /**
1844      * @param etiquetaGuarderiaBorrada
1845      */
1846     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1847         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1848
1849         if (guarderia != null) {
1850             ss.delete(guarderia);
1851
1852             tst.commit();
1853         }
1854     }
1855
1856     /**
1857      * @param etiquetaGuarderiaBorrada
1858      */
1859     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1860         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1861
1862         if (guarderia != null) {
1863             guarderia.setSalario(salario);
1864
1865             ss.update(guarderia);
1866
1867             tst.commit();
1868         }
1869     }
1870
1871     /**
1872      * @param etiquetaGuarderiaBorrada
1873      */
1874     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1875         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1876
1877         if (guarderia != null) {
1878             ss.delete(guarderia);
1879
1880             tst.commit();
1881         }
1882     }
1883
1884     /**
1885      * @param etiquetaGuarderiaBorrada
1886      */
1887     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1888         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1889
1890         if (guarderia != null) {
1891             guarderia.setSalario(salario);
1892
1893             ss.update(guarderia);
1894
1895             tst.commit();
1896         }
1897     }
1898
1899     /**
1900      * @param etiquetaGuarderiaBorrada
1901      */
1902     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1903         Guarderia guarderia = new Guarderia(codigoGuarderia);
1904
1905         if (guarderia != null) {
1906             ss.save(guarderia);
1907
1908             tst.commit();
1909         }
1910     }
1911
1912     /**
1913      * @param etiquetaGuarderiaBorrada
1914      */
1915     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
1916         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1917
1918         if (guarderia != null) {
1919             guarderia.setNombre(nombre);
1920
1921             ss.update(guarderia);
1922
1923             tst.commit();
1924         }
1925     }
1926
1927     /**
1928      * @param etiquetaGuarderiaBorrada
1929      */
1930     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1931         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1932
1933         if (guarderia != null) {
1934             ss.delete(guarderia);
1935
1936             tst.commit();
1937         }
1938     }
1939
1940     /**
1941      * @param etiquetaGuarderiaBorrada
1942      */
1943     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1944         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1945
1946         if (guarderia != null) {
1947             guarderia.setSalario(salario);
1948
1949             ss.update(guarderia);
1950
1951             tst.commit();
1952         }
1953     }
1954
1955     /**
1956      * @param etiquetaGuarderiaBorrada
1957      */
1958     public void borrarGuarderia(Session ss, String codigoGuarderia) {
1959         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1960
1961         if (guarderia != null) {
1962             ss.delete(guarderia);
1963
1964             tst.commit();
1965         }
1966     }
1967
1968     /**
1969      * @param etiquetaGuarderiaBorrada
1970      */
1971     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
1972         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
1973
1974         if (guarderia != null) {
1975             guarderia.setSalario(salario);
1976
1977             ss.update(guarderia);
1978
1979             tst.commit();
1980         }
1981     }
1982
1983     /**
1984      * @param etiquetaGuarderiaBorrada
1985      */
1986     public void agregarGuarderia(Session ss, String codigoGuarderia) {
1987         Guarderia guarderia = new Guarderia(codigoGuarderia);
1988
1989         if (guarderia != null) {
1990             ss.save(guarderia);
1991
1992             tst.commit();
1993         }
1994     }
1995
1996     /**
1997      * @param etiquetaGuarderiaBorrada
1998      */
1999     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2000         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2001
2002         if (guarderia != null) {
2003             guarderia.setNombre(nombre);
2004
2005             ss.update(guarderia);
2006
2007             tst.commit();
2008         }
2009     }
2010
2011     /**
2012      * @param etiquetaGuarderiaBorrada
2013      */
2014     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2015         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2016
2017         if (guarderia != null) {
2018             ss.delete(guarderia);
2019
2020             tst.commit();
2021         }
2022     }
2023
2024     /**
2025      * @param etiquetaGuarderiaBorrada
2026      */
2027     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2028         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2029
2030         if (guarderia != null) {
2031             guarderia.setSalario(salario);
2032
2033             ss.update(guarderia);
2034
2035             tst.commit();
2036         }
2037     }
2038
2039     /**
2040      * @param etiquetaGuarderiaBorrada
2041      */
2042     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2043         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2044
2045         if (guarderia != null) {
2046             ss.delete(guarderia);
2047
2048             tst.commit();
2049         }
2050     }
2051
2052     /**
2053      * @param etiquetaGuarderiaBorrada
2054      */
2055     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2056         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2057
2058         if (guarderia != null) {
2059             guarderia.setSalario(salario);
2060
2061             ss.update(guarderia);
2062
2063             tst.commit();
2064         }
2065     }
2066
2067     /**
2068      * @param etiquetaGuarderiaBorrada
2069      */
2070     public void agregarGuarderia(Session ss, String codigoGuarderia) {
2071         Guarderia guarderia = new Guarderia(codigoGuarderia);
2072
2073         if (guarderia != null) {
2074             ss.save(guarderia);
2075
2076             tst.commit();
2077         }
2078     }
2079
2080     /**
2081      * @param etiquetaGuarderiaBorrada
2082      */
2083     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2084         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2085
2086         if (guarderia != null) {
2087             guarderia.setNombre(nombre);
2088
2089             ss.update(guarderia);
2090
2091             tst.commit();
2092         }
2093     }
2094
2095     /**
2096      * @param etiquetaGuarderiaBorrada
2097      */
2098     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2099         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2100
2101         if (guarderia != null) {
2102             ss.delete(guarderia);
2103
2104             tst.commit();
2105         }
2106     }
2107
2108     /**
2109      * @param etiquetaGuarderiaBorrada
2110      */
2111     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2112         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2113
2114         if (guarderia != null) {
2115             guarderia.setSalario(salario);
2116
2117             ss.update(guarderia);
2118
2119             tst.commit();
2120         }
2121     }
2122
2123     /**
2124      * @param etiquetaGuarderiaBorrada
2125      */
2126     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2127         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2128
2129         if (guarderia != null) {
2130             ss.delete(guarderia);
2131
2132             tst.commit();
2133         }
2134     }
2135
2136     /**
2137      * @param etiquetaGuarderiaBorrada
2138      */
2139     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2140         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2141
2142         if (guarderia != null) {
2143             guarderia.setSalario(salario);
2144
2145             ss.update(guarderia);
2146
2147             tst.commit();
2148         }
2149     }
2150
2151     /**
2152      * @param etiquetaGuarderiaBorrada
2153      */
2154     public void agregarGuarderia(Session ss, String codigoGuarderia) {
2155         Guarderia guarderia = new Guarderia(codigoGuarderia);
2156
2157         if (guarderia != null) {
2158             ss.save(guarderia);
2159
2160             tst.commit();
2161         }
2162     }
2163
2164     /**
2165      * @param etiquetaGuarderiaBorrada
2166      */
2167     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2168         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2169
2170         if (guarderia != null) {
2171             guarderia.setNombre(nombre);
2172
2173             ss.update(guarderia);
2174
2175             tst.commit();
2176         }
2177     }
2178
2179     /**
2180      * @param etiquetaGuarderiaBorrada
2181      */
2182     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2183         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2184
2185         if (guarderia != null) {
2186             ss.delete(guarderia);
2187
2188             tst.commit();
2189         }
2190     }
2191
2192     /**
2193      * @param etiquetaGuarderiaBorrada
2194      */
2195     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2196         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2197
2198         if (guarderia != null) {
2199             guarderia.setSalario(salario);
2200
2201             ss.update(guarderia);
2202
2203             tst.commit();
2204         }
2205     }
2206
2207     /**
2208      * @param etiquetaGuarderiaBorrada
2209      */
2210     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2211         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2212
2213         if (guarderia != null) {
2214             ss.delete(guarderia);
2215
2216             tst.commit();
2217         }
2218     }
2219
2220     /**
2221      * @param etiquetaGuarderiaBorrada
2222      */
2223     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2224         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2225
2226         if (guarderia != null) {
2227             guarderia.setSalario(salario);
2228
2229             ss.update(guarderia);
2230
2231             tst.commit();
2232         }
2233     }
2234
2235     /**
2236      * @param etiquetaGuarderiaBorrada
2237      */
2238     public void agregarGuarderia(Session ss, String codigoGuarderia) {
2239         Guarderia guarderia = new Guarderia(codigoGuarderia);
2240
2241         if (guarderia != null) {
2242             ss.save(guarderia);
2243
2244             tst.commit();
2245         }
2246     }
2247
2248     /**
2249      * @param etiquetaGuarderiaBorrada
2250      */
2251     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2252         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2253
2254         if (guarderia != null) {
2255             guarderia.setNombre(nombre);
2256
2257             ss.update(guarderia);
2258
2259             tst.commit();
2260         }
2261     }
2262
2263     /**
2264      * @param etiquetaGuarderiaBorrada
2265      */
2266     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2267         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2268
2269         if (guarderia != null) {
2270             ss.delete(guarderia);
2271
2272             tst.commit();
2273         }
2274     }
2275
2276     /**
2277      * @param etiquetaGuarderiaBorrada
2278      */
2279     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2280         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2281
2282         if (guarderia != null) {
2283             guarderia.setSalario(salario);
2284
2285             ss.update(guarderia);
2286
2287             tst.commit();
2288         }
2289     }
2290
2291     /**
2292      * @param etiquetaGuarderiaBorrada
2293      */
2294     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2295         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2296
2297         if (guarderia != null) {
2298             ss.delete(guarderia);
2299
2300             tst.commit();
2301         }
2302     }
2303
2304     /**
2305      * @param etiquetaGuarderiaBorrada
2306      */
2307     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2308         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2309
2310         if (guarderia != null) {
2311             guarderia.setSalario(salario);
2312
2313             ss.update(guarderia);
2314
2315             tst.commit();
2316         }
2317     }
2318
2319     /**
2320      * @param etiquetaGuarderiaBorrada
2321      */
2322     public void agregarGuarderia(Session ss, String codigoGuarderia) {
2323         Guarderia guarderia = new Guarderia(codigoGuarderia);
2324
2325         if (guarderia != null) {
2326             ss.save(guarderia);
2327
2328             tst.commit();
2329         }
2330     }
2331
2332     /**
2333      * @param etiquetaGuarderiaBorrada
2334      */
2335     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2336         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2337
2338         if (guarderia != null) {
2339             guarderia.setNombre(nombre);
2340
2341             ss.update(guarderia);
2342
2343             tst.commit();
2344         }
2345     }
2346
2347     /**
2348      * @param etiquetaGuarderiaBorrada
2349      */
2350     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2351         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2352
2353         if (guarderia != null) {
2354             ss.delete(guarderia);
2355
2356             tst.commit();
2357         }
2358     }
2359
2360     /**
2361      * @param etiquetaGuarderiaBorrada
2362      */
2363     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2364         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2365
2366         if (guarderia != null) {
2367             guarderia.setSalario(salario);
2368
2369             ss.update(guarderia);
2370
2371             tst.commit();
2372         }
2373     }
2374
2375     /**
2376      * @param etiquetaGuarderiaBorrada
2377      */
2378     public void borrarGuarderia(Session ss, String codigoGuarderia) {
2379         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2380
2381         if (guarderia != null) {
2382             ss.delete(guarderia);
2383
2384             tst.commit();
2385         }
2386     }
2387
2388     /**
2389      * @param etiquetaGuarderiaBorrada
2390      */
2391     public void actualizarSalarioGuarderia(Session ss, String codigoGuarderia, Double salario) {
2392         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);
2393
2394         if (guarderia != null) {
2395             guarderia.setSalario(salario);
2396
2397             ss.update(guarderia);
2398
2399             tst.commit();
2400         }
2401     }
2402
2403     /**
2404      * @param etiquetaGuarderiaBorrada
2405      */
2406     public void agregarGuarderia(Session ss, String codigoGuarderia) {
2407         Guarderia guarderia = new Guarderia(codigoGuarderia);
2408
2409         if (guarderia != null) {
2410             ss.save(guarderia);
2411
2412             tst.commit();
2413         }
2414     }
2415
2416     /**
2417      * @param etiquetaGuarderiaBorrada
2418      */
2419     public void actualizarGuarderia(Session ss, String codigoGuarderia, String nombre) {
2420         Guarderia guarderia = (Guarderia) ss.get(Guarderia.class, codigoGuarderia);

```

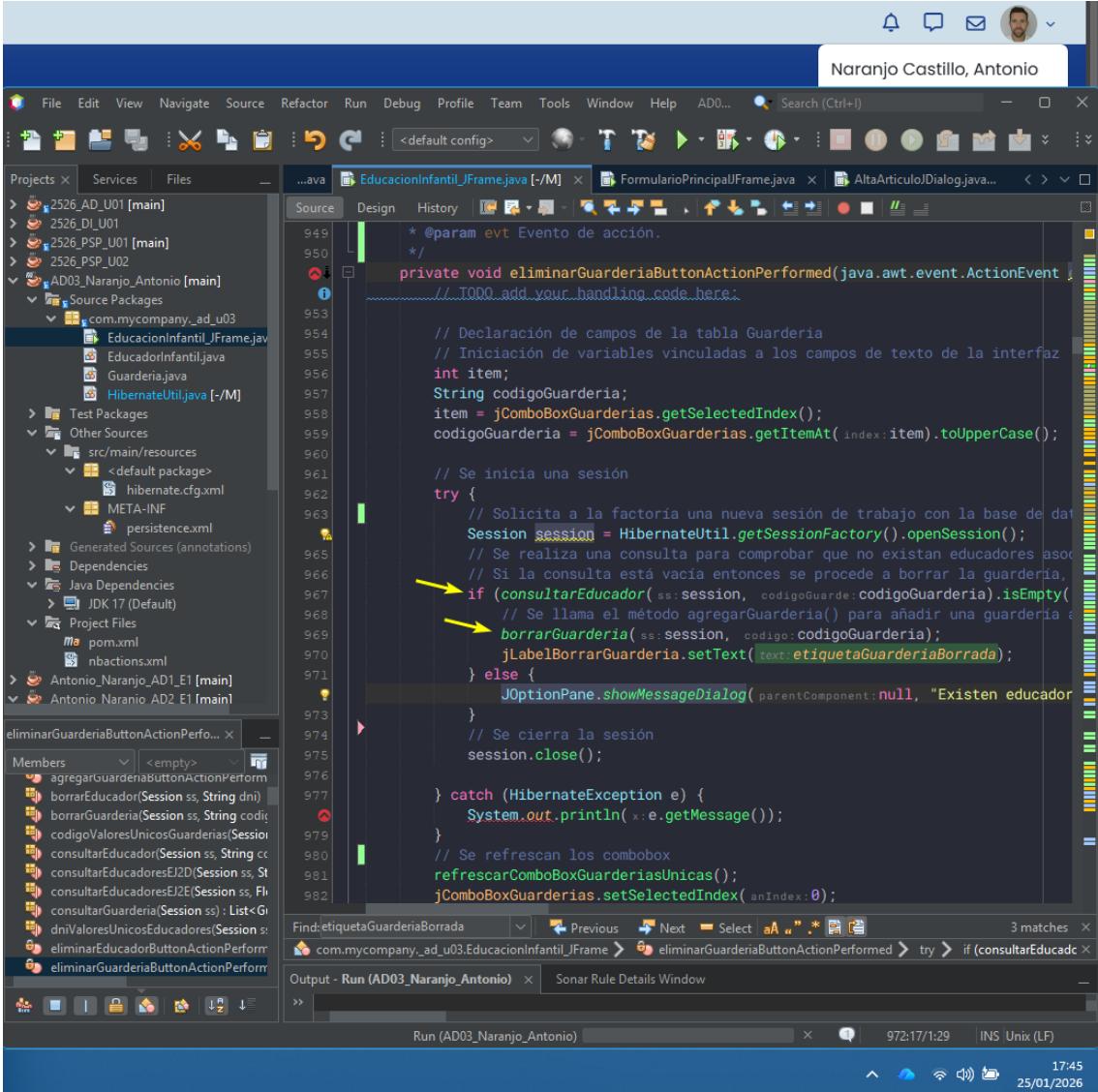
b) Eliminar guarderías y educadores infantiles de la base de datos. Si una guardería tiene educadores asignados no se podrá eliminar la guardería.

The screenshot shows a Java Swing application window titled "Gestión de Centros de Educación Infantil". The window has several tabs:

- 1) Añadir Objetos**: Contains forms for adding a "Guarderías" (with fields: Código: HJK05, Nombre: EL CASTILLO DE MI BEBÉ, Capacidad: 50, Presupuesto: 35000.0) and an "Educador" (with fields: DNI: 87654321Z, Nombre: ALBA, Apellidos: ÁLVAREZ GONZÁLEZ, Código Guardería: HJK05). Success messages are displayed below each addition.
- 2) Eliminar Objetos**: Contains forms for deleting a "Guarderías" (with Código: HJK05 selected) and an "Educador" (with DNI: 11111111A selected). A red box highlights the "Borrar" button for the kindergarten.
- 3) Listar Guarderías**: Shows a list of kindergartens.
- 5) Listar Educadores**: Shows a list of educators.
- 6) Actualizar salario Educador Infantil**: Shows fields for updating an educator's salary (DNI: 11111111A, Nuevo Salario).

A central modal dialog box is displayed, titled "Message", with the message: "Existen educadores asignados a la guardería: HJK05. La guardería no se puede borrar." (There are educators assigned to the kindergarten: HJK05. The kindergarten cannot be deleted.) An "OK" button is at the bottom right of the dialog, with a red arrow pointing to it labeled "3".

Se selecciona el código de una guardería (se selecciona la recientemente creada) y se intenta borrar pulsando sobre el botón borrar, como existe un educador asignado a dicha guardería no permite su borrado mostrando en pantalla a modo de mensaje informativo JOptionPane.showMessageDialog.



```

private void eliminarGuarderiaButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

// Declaración de campos de la tabla Guarderia
// Iniciación de variables vinculadas a los campos de texto de la interfaz
int item;
String codigoGuarderia;
item = jComboBoxGuarderias.getSelectedIndex();
codigoGuarderia = jComboBoxGuarderias.getItemAt(item).toUpperCase();

// Se inicia una sesión
try {
    // Solicitud a la factoría una nueva sesión de trabajo con la base de datos
    Session session = HibernateUtil.getSessionFactory().openSession();
    // Se realiza una consulta para comprobar que no existan educadores asociados
    // Si la consulta está vacía entonces se procede a borrar la guardería
    if (consultarEducador(ss, session, codigoGuarderia).isEmpty()) {
        // Se llama el método agregarGuarderia() para añadir una guardería a la lista
        borrarGuarderia(ss, session, codigoGuarderia);
        jLabelBorrarGuarderia.setText("etiquetaGuarderiaBorrada");
    } else {
        JOptionPane.showMessageDialog(parentComponent, "Existen educadores");
    }
    // Se cierra la sesión
    session.close();
} catch (HibernateException e) {
    System.out.println(e.getMessage());
}
// Se refrescan los combobox
refrescarComboBoxGuarderiasUnicas();
jComboBoxGuarderias.setSelectedIndex(0);

```

Find:etiquetaGuarderiaBorrada Previous Next Select aA " * Output - Run (AD03_Naranjo_Antonio) 972:17/1:29 17:45 25/01/2026

El método del botón borrar del apartado Guarderías de la interfaz gráfica swing, toma el valor del ComboBox seleccionado previamente por el usuario, crea un objeto Session, realiza una consulta ejecutando para ello el método consultarEducador() pasando como argumento el objeto Session y el código de la guardería (PK) mediante el cual se comprobará si existe algún educador asociado a la guardería que se pretende borrar, si el método devuelve una lista vacía se procede a borrar la guardería ejecutando el método borrarGuarderia() (mismo argumentos que el método anterior) en caso contrario muestra el mensaje por pantalla de su imposibilidad de borrarse.

The screenshot shows the NetBeans IDE interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, AD0..., Search (Ctrl+F).
- User Information:** Naranjo Castillo, Antonio.
- Projects Tab:** Shows the project structure:
 - 2526_AD_U01 [main]
 - 2526_DL_U01
 - 2526_PSP_U01 [main]
 - 2526_PSP_U02
 - AD03_Naranjo_Antonio [main]
 - Source Packages
 - com.mycompany._ad_u03
 - EducacionInfantil.JFrame.java
 - EducadorInfantil.java
 - Guarderia.java
 - HibernateUtil.java
 - Test Packages
 - Other Sources
 - src/main/resources
 - hibernate.cfg.xml
 - META-INF
 - persistence.xml
 - Generated Sources (annotations)
 - Dependencies
 - Java Dependencies
 - JDK 17 (Default)
 - Project Files
 - pom.xml
 - nbactions.xml
- Antonio_Naranjo_AD1_E1 [main]
- Antonio_Naranjo_AD2_E1 [main]

- Code Editor:** The current file is `EducacionInfantil.JFrame.java`. The code implements a method `consultarEducador` which performs a database query to find educators by their ID. The code includes annotations for Hibernate and Java Persistence API (JPA). A yellow arrow points to the line where the query is created: `Query<EducadorInfantil> query = ss.createQuery(string: "from EducadorInfantil where id = :cod")`.
- Navigator:** Shows the members of the class, including methods like `agregarGuarderiaButtonActionPerform`, `borrarGuarderia(Session ss, String dni)`, and `consultarGuarderia(Session ss) : List<Guarderia>`.
- Output:** Shows the output of the run configuration for `AD03_Naranjo_Antonio`.

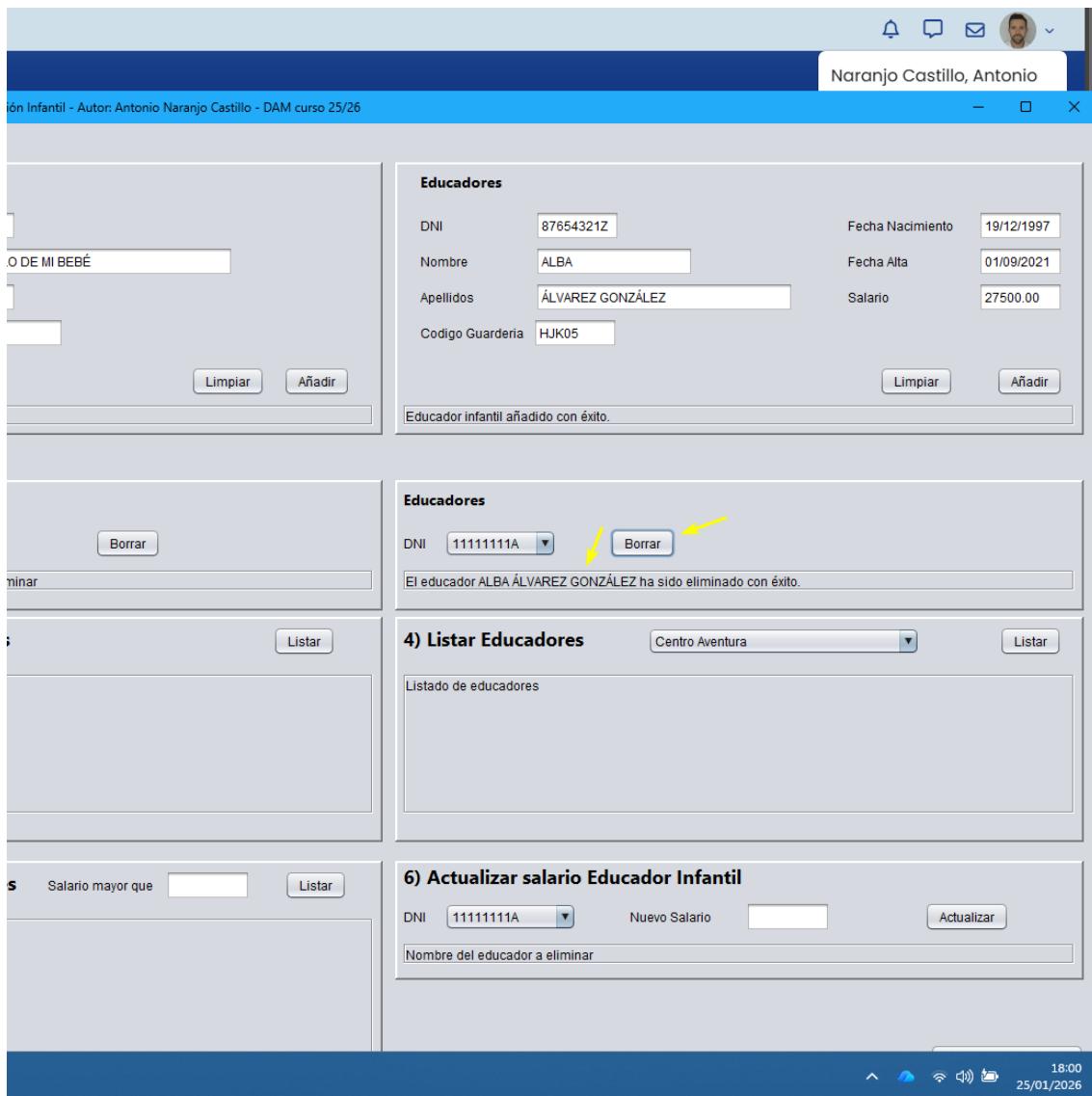
El método consultarEducador() que recibe como argumentos el objeto Session y el String código de la guardería, crea un objeto Transaction, crea una consulta Query de objetos EducadorInfantil, se establecen el código de la guardería como parámetros de entrada de la consulta y se recoge la lista de objetos EducadoresInfantil que será devuelta por el método.

```

EducacionInfantil.JFrame.java [-/M]
...
1401     static void borrarGuarderia(Session ss, String codigo) {
1402
1403         // Inicia la transacción para asegurar que el borrado sea atómico y seguro
1404         Transaction tst = ss.beginTransaction();
1405
1406         try {
1407
1408             // Intenta recuperar el objeto Guarderia de la BD usando su clase y la clave
1409             Guarderia guarde = ss.get(type:Guarderia.class, srlzbl:codigo);
1410
1411             // Verifica si el objeto no existe
1412             if (guarde == null) {
1413                 System.out.println("No se encontró la guardería con código: " + codigo);
1414             } else {
1415
1416                 // Ordena a Hibernate marcar el objeto persistente para su eliminación
1417                 ss.delete(:guarde);
1418                 System.out.println("Se borró la guardería con código: " + codigo);
1419                 // Almacena un mensaje de éxito con el nombre real de la guardería
1420                 etiquetaGuarderiaBorrada = "La guardería " + guarde.getNombre() + " ha sido borrada";
1421             }
1422
1423             // Ejecuta el commit para aplicar definitivamente el borrado en las tablas
1424             tst.commit();
1425
1426         } catch (Exception e) {
1427
1428             if (tst != null) {
1429                 // Hacer rollback en caso de error
1430                 tst.rollback();
1431             }
1432             System.out.println( e.getMessage());
1433         }
1434     }
1435
1436     ...

```

El método `borrarGuarderia()` recibe como argumentos el objeto `Session` y el `String` código de la guardería, crea un objeto `Transaction`, crea el objeto `Guarderia` ejecutando el método `get()` del objeto `Session` pasando como argumento el `String` anterior código guardería, si el objeto no es nulo se procede a confirmar su persistencia ejecutando el método `delete()` del objeto `Session`, posteriormente, se ejecuta el método `commit()` del objeto `Transaction` para confirmar los cambios. Si se produce algún error se capta y se ejecuta el método `rollback()` para no confirmar los cambios.



Se procede a borrar el reciente educador que se creó en el apartado anterior, se aplica sobre el botón Borrar del apartado Educadores y se presenta una etiqueta con el resultado obtenido, y el posterior refresco del objeto ComboBox (de ahí que se muestre el primero de la lista porque así se ha decidido a nivel de código).

```

git-cmd - \mysqld
Símbolo del sistema - \mysql
Naranjo Castillo, Antonio

+-----+
| ABC01 | Centro Aventura      |      50 |    20000 |
| ABC02 | Los arrecifes          |      70 |    45000 |
| CDE03 | Barcos de Papel        |      30 |    15000 |
| DFG04 | La casita de Noa        |      40 |    40000 |
| HJK05 | EL CASTILLO DE MI BEBÉ |      50 |    35000 |
+-----+
5 rows in set (0.005 sec)

mysql> SELECT * FROM educador_infantil;
+-----+
| dni   | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+
| 10101010L | ELISA  | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000  | DFG04
| 11111111A | PEPE   | LOPEZ MARIN   | 1984-03-02 | 2010-03-01 | 35000  | ABC01
| 22222222B | LOLA   | PEREZ PEREZ  | 1988-01-01 | 2010-03-01 | 32000  | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO  | 1996-03-03 | 2015-01-01 | 25000  | ABC01
| 44444444D | LAURA  | GOMEZ GOMEZ  | 1978-01-20 | 2010-08-01 | 40000  | ABC02
| 55555555F | JUAN   | LOPEZ LOPEZ  | 1988-05-01 | 2008-05-03 | 25000  | ABC02
| 66666666G | ANA    | MARIN MARIN  | 2000-03-02 | 2020-09-01 | 18000  | ABC02
| 77777777H | CARMEN | CASAS CAVA   | 1975-09-25 | 2000-05-08 | 35000  | CDE03
| 87654321Z | ALBA   | ALVAREZ GONZALEZ | 1997-12-19 | 2021-09-01 | 27500  | HJK05
| 88888888J | MARCOS | LUNA PIO     | 1989-08-15 | 2010-01-01 | 38000  | CDE03
| 99999999K | JUANA  | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500  | DFG04
+-----+
11 rows in set (0.013 sec)

mysql> SELECT * FROM educador_infantil;
+-----+
| dni   | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+
| 10101010L | ELISA  | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000  | DFG04
| 11111111A | PEPE   | LOPEZ MARIN   | 1984-03-02 | 2010-03-01 | 35000  | ABC01
| 22222222B | LOLA   | PEREZ PEREZ  | 1988-01-01 | 2010-03-01 | 32000  | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO  | 1996-03-03 | 2015-01-01 | 25000  | ABC01
| 44444444D | LAURA  | GOMEZ GOMEZ  | 1978-01-20 | 2010-08-01 | 40000  | ABC02
| 55555555F | JUAN   | LOPEZ LOPEZ  | 1988-05-01 | 2008-05-03 | 25000  | ABC02
| 66666666G | ANA    | MARIN MARIN  | 2000-03-02 | 2020-09-01 | 18000  | ABC02
| 77777777H | CARMEN | CASAS CAVA   | 1975-09-25 | 2000-05-08 | 35000  | CDE03
| 88888888J | MARCOS | LUNA PIO     | 1989-08-15 | 2010-01-01 | 38000  | CDE03
| 99999999K | JUANA  | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500  | DFG04
+-----+
10 rows in set (0.011 sec)

mysql> |

```

Se muestran en la consola mysql la eliminación del educador infantil en cuestión.

1) Añadir Objetos

Guarderías		Educadores	
Código	HJK05	DNI	87654321Z
Nombre	EL CASTILLO DE MI BEBÉ	Nombre	ALBA
Capacidad	50	Apellidos	ÁLVAREZ GONZÁLEZ
Presupuesto	35000.0	Código Guardería	HJK05

Guardería añadida con éxito.

Educador infantil añadido con éxito.

2) Eliminar Objetos

Guarderías		Educadores	
Código	ABC01	DNI	11111111A
Borrar		Borrar	
La guardería EL CASTILLO DE MI BEBÉ ha sido eliminada con éxito.			

El educador ALBA ÁLVAREZ GONZÁLEZ ha sido eliminado con éxito.

3) Listar Guarderías

Listado de guarderías

4) Listar Educadores

Centro Aventura

Listado de educadores

5) Listar Educadores

Salario mayor que Listar

Listado de educadores

6) Actualizar salario Educador Infantil

DNI Nuevo Salario
Nombre del educador a eliminar

18:04
25/01/2026

Ahora sí, una vez eliminado el educador asociado sí permite borrar la guardería que se pretendía eliminar. Al igual que ocurría en el paso anterior, se refrescan los valores del ComboBox y se apunta al primero de la lista.

```

git-cmd - \mysql
Símbolo del sistema - \mysql
+-----+-----+-----+-----+-----+-----+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+-----+-----+-----+-----+-----+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 40000 | ABC02
| 55555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03
| 87654321Z | ALBA | ÁLVAREZ GONZÁLEZ | 1997-12-19 | 2021-09-01 | 27500 | HJK05
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03
| 99999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04
+-----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.013 sec)

mysql> SELECT * FROM educador_infantil;
+-----+-----+-----+-----+-----+-----+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+-----+-----+-----+-----+-----+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 40000 | ABC02
| 55555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03
| 99999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04
+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.011 sec)

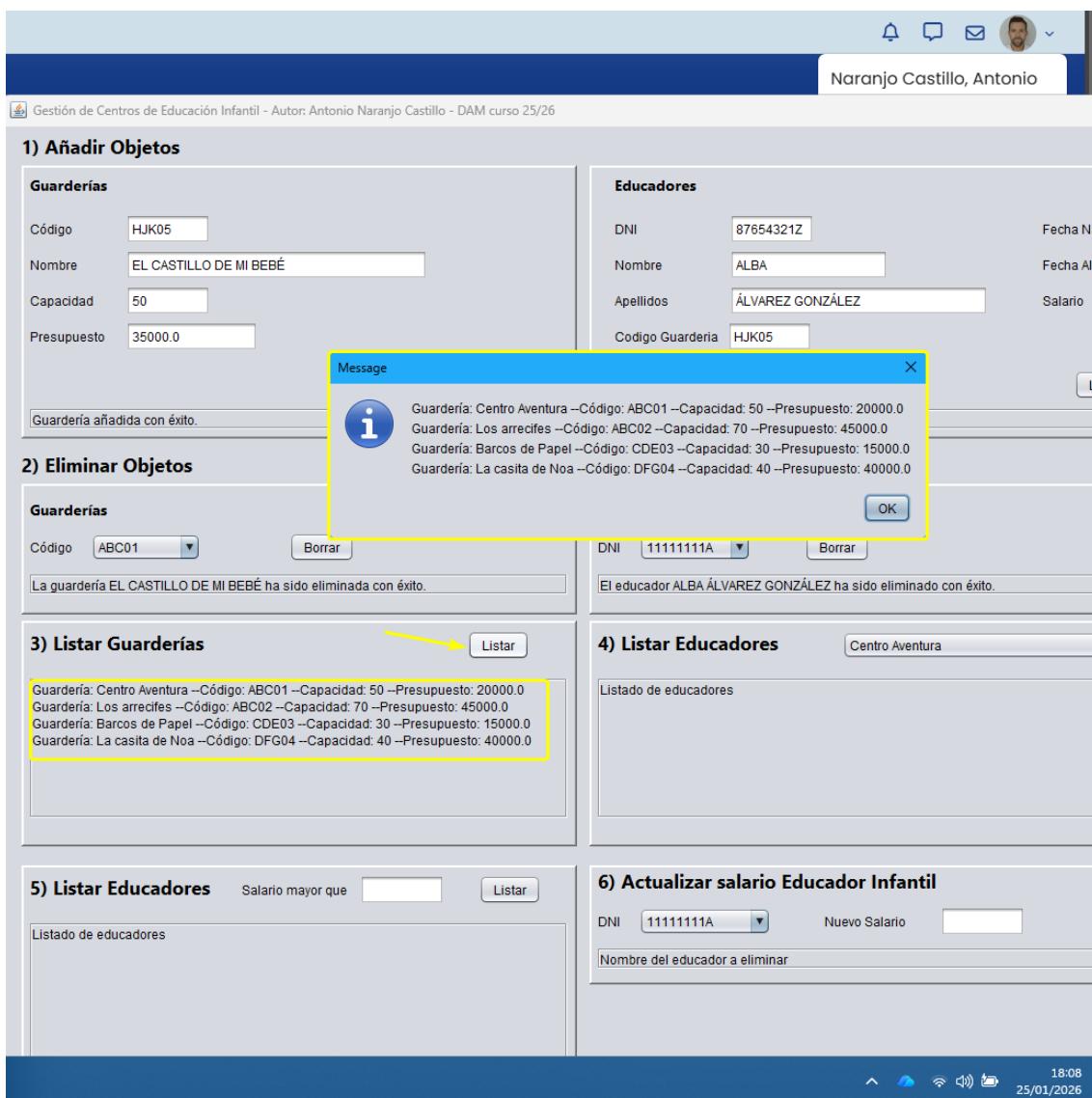
mysql> SELECT * FROM guarderia;
+-----+-----+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+-----+-----+
| ABC01 | Centro Aventura | 50 | 20000 |
| ABC02 | Los arrecifes | 70 | 45000 |
| CDE03 | Barcos de Papel | 30 | 15000 |
| DFG04 | La casita de Noa | 40 | 40000 |
+-----+-----+-----+
4 rows in set (0.012 sec)

mysql> |

```

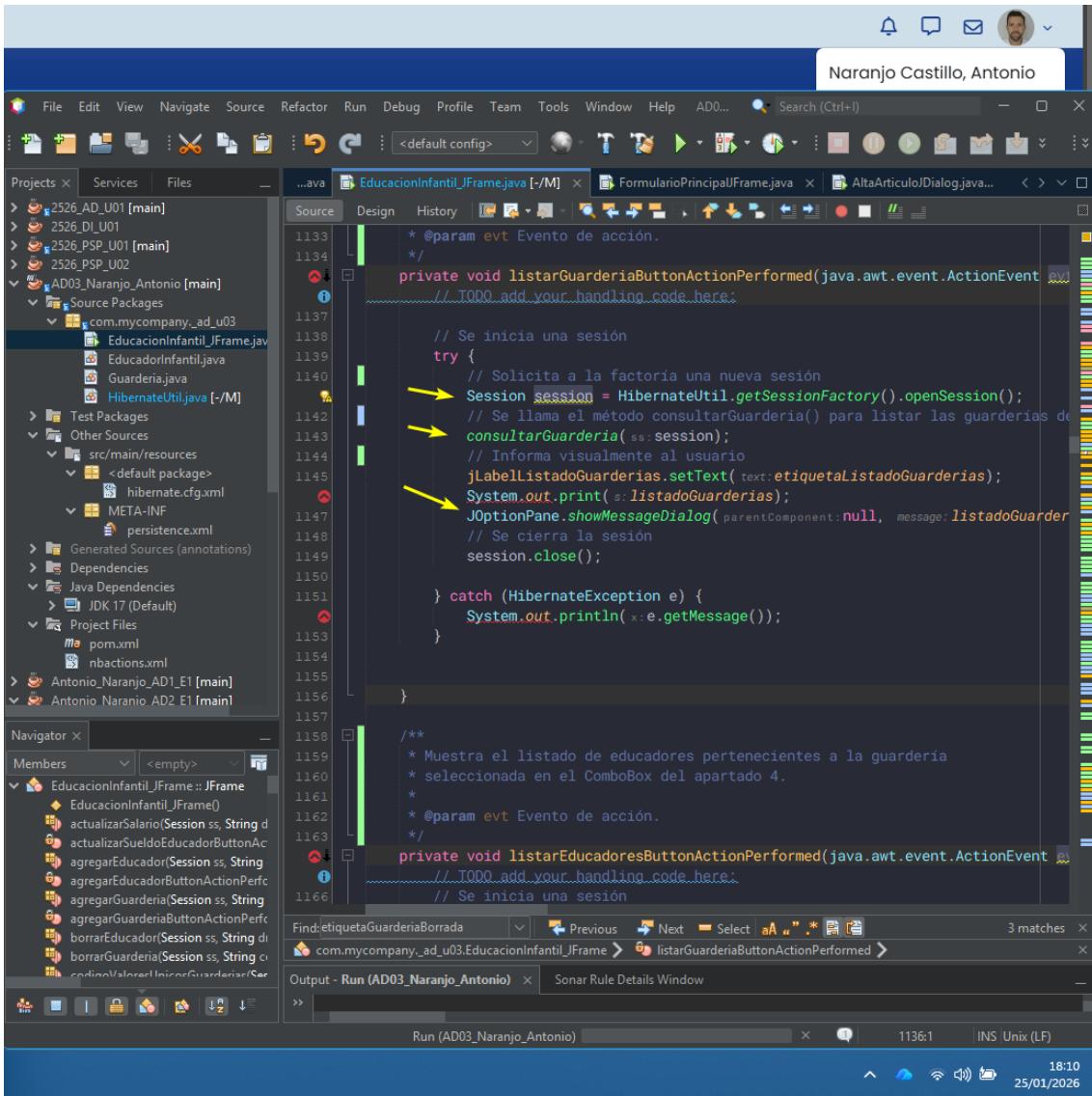
Se muestran los nuevos datos de la tabla guarderia, y ya no aparecen los datos de la guardería que se añadió en el apartado anterior.

c) Mostrar el listado de todas las guarderías.



Se pulsa sobre el botón Listar del apartado Listar Guarderías mostrándose el listado de guardería tanto en una etiqueta objeto JLabel como en un objeto JOptionPane. La alimentación del combobox se lleva a cabo previamente realizando una consulta de valores únicos de objetos Guarderia por medio de su atributo código, aunque no es del todo necesario porque dicho atributo es clave primaria, aunque se considera una buena práctica, se ejecuta el método refreshComboBoxGuarderiasUnicas().

Los métodos que refrescan los valores de los ítems de los combobox se ejecutan dentro del método constructor de la clase de la interfaz gráfica EducacionInfantil_JFrame.



```

private void listarGuarderiaButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // Se inicia una sesión
    try {
        // Sigue la ejecución de la clase anterior
        Session session = HibernateUtil.getSessionFactory().openSession();
        consultarGuarderia(ss, session);
        // Informa visualmente al usuario
        jLabelListadoGuarderias.setText("etiquetaListadoGuarderias");
        System.out.println(ss.listadoGuarderias);
        JOptionPane.showMessageDialog(null, "listadoGuarderias");
        // Se cierra la sesión
        session.close();
    } catch (HibernateException e) {
        System.out.println(e.getMessage());
    }
}

/**
 * Muestra el listado de educadores pertenecientes a la guardería
 * seleccionada en el ComboBox del apartado 4.
 */
* @param evt Evento de acción.
*/
private void listarEducadoresButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // Se inicia una sesión
}

```

The screenshot shows the NetBeans IDE interface with the code editor open. The code is written in Java and uses Hibernate for database interaction. Three yellow arrows point to the line `Session session = HibernateUtil.getSessionFactory().openSession();` to highlight it. The code is part of a button's action listener.

El botón Listar ejecuta un método que crea un objeto Session, ejecuta el método consultarGuarderia() pasándole como argumento el objeto Session anterior, del cual se obtiene una lista de objetos Guarderia. Posteriormente, se muestran los datos obtenidos a partir de una variable estática de la clase principal de la interfaz swing EducacionInfantil_JFrame. Esta variable estática se alimenta en el mismo método consultarGuarderia().

```

static List<Guarderia> consultarGuarderia(Session ss) {
    Transaction tst = ss.beginTransaction();
    List<Guarderia> guarderias = new ArrayList<>();
    StringBuilder sb = new StringBuilder();
    StringBuilder sbHTML = new StringBuilder(str."<html>");
    String texto;

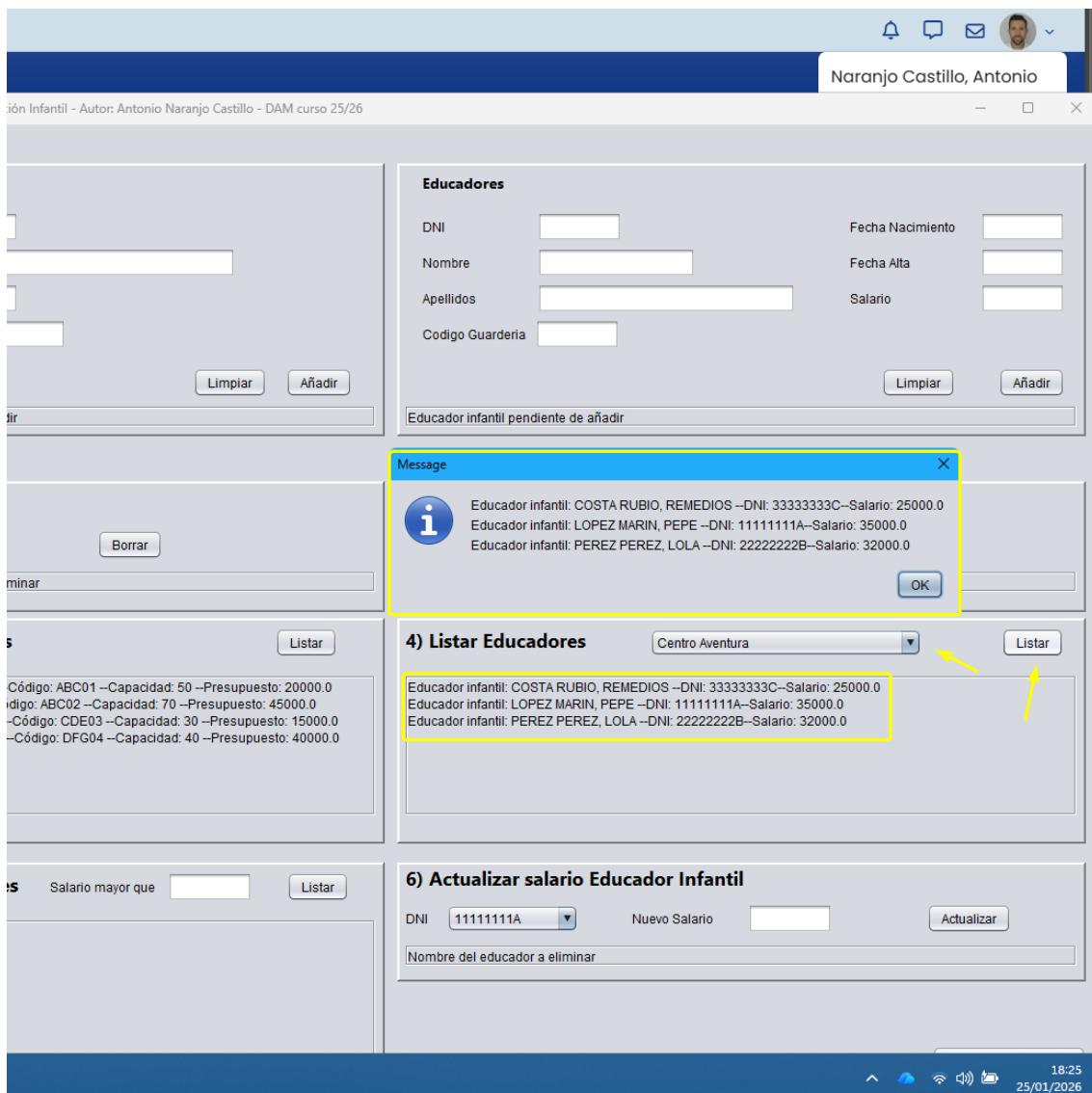
    try {
        // Crea la consulta HQL
        Query<Guarderia> query = ss.createQuery("from Guarderia g", type);
        // Ejecuta la consulta y vuelca los resultados en la lista
        guarderias = query.list();

        // Si la lista no está vacía se almacenan los datos para presentarlos
        if (!guarderias.isEmpty()) {
            for (Guarderia guarde : guarderias) {
                texto = "Guardería: " + guarde.getNombre() + " --Código: " + guarde.getId();
                System.out.println(x:texto);
                sb.append(str:texto).append(str:\n");
                sbHTML.append(str:texto).append(str:<br>");
            }
            sbHTML.append(str:</html>");
            listadoGuarderias = sb.toString();
            etiquetaListadoGuarderias = sbHTML.toString();
        } else {
            texto = "No existen guarderías que listar";
            listadoGuarderias = texto;
            etiquetaListadoGuarderias = texto;
        }
        // Se confirma la transacción
        tst.commit();
    } catch (Exception e) {
        tst.rollback();
    }
}

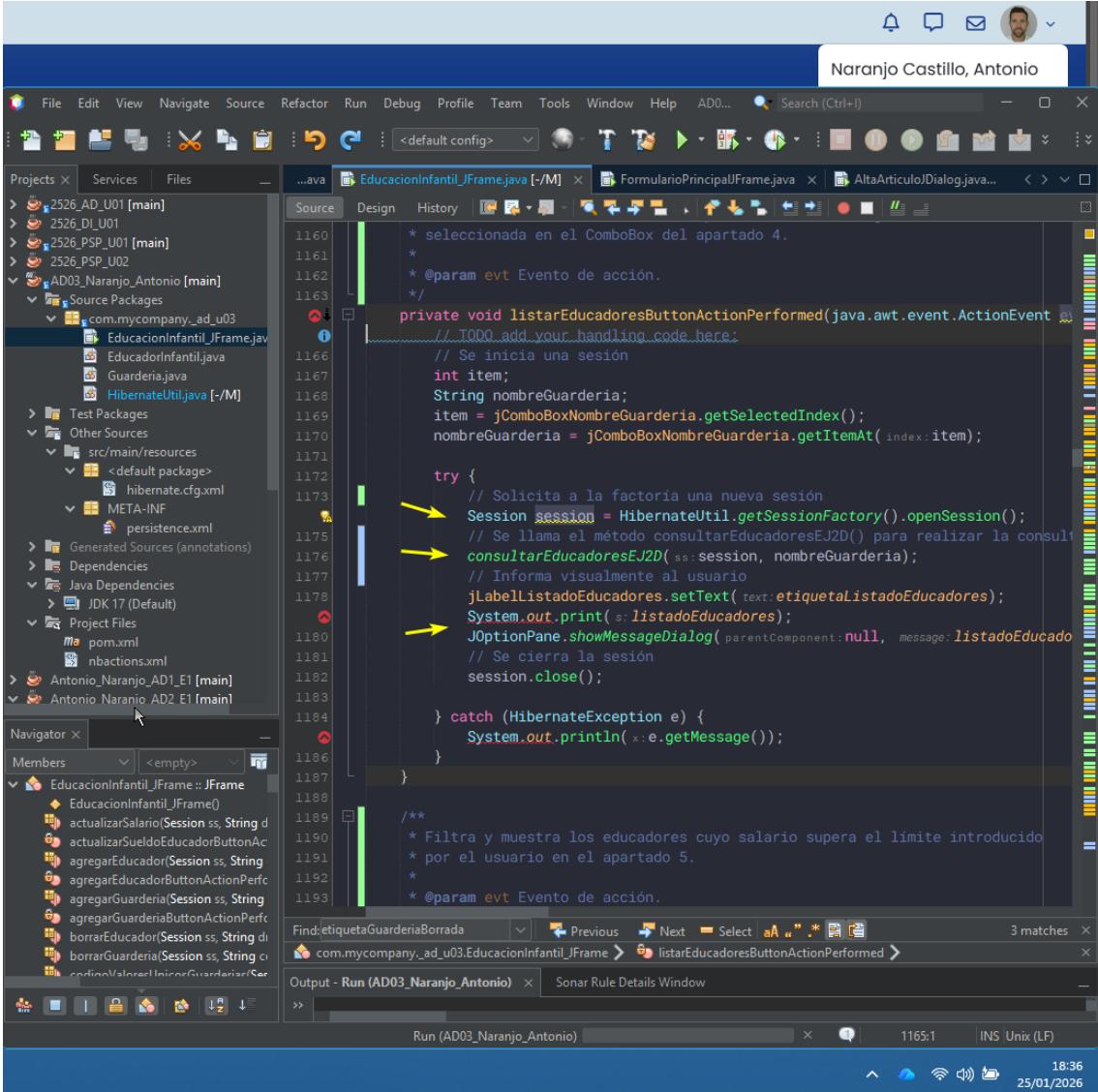
```

En el método `consultarGuarderia()` se le pasa como argumento el objeto `Session` necesario para crear el objeto `Transaction`. Luego se crea una lista `Query` de objetos `Guarderia` tras realizar la consulta de todas las guarderías disponibles en la tabla `guarderia`, mediante el método `list()` se almacena la consulta en una lista de `Guarderías`. Se comprueba que la lista no esté vacía para proceder con la impresión de los resultados, tanto por medio de una etiqueta en la interfaz como empleando un objeto `StringBuilder` el cual se puede formatear mediante HTML y mostrar su resultado en un objeto `JOPTIONPANE`. Por último, se ejecuta el método `commit()` del objeto `Transaction` para confirmar los cambios o en caso de error se ejecuta `rollback()` tras capturar el error.

d) Mostrar un listado con todos los educadores infantiles (ordenados por apellidos) de una guardería indicada por el usuario mediante su nombre.



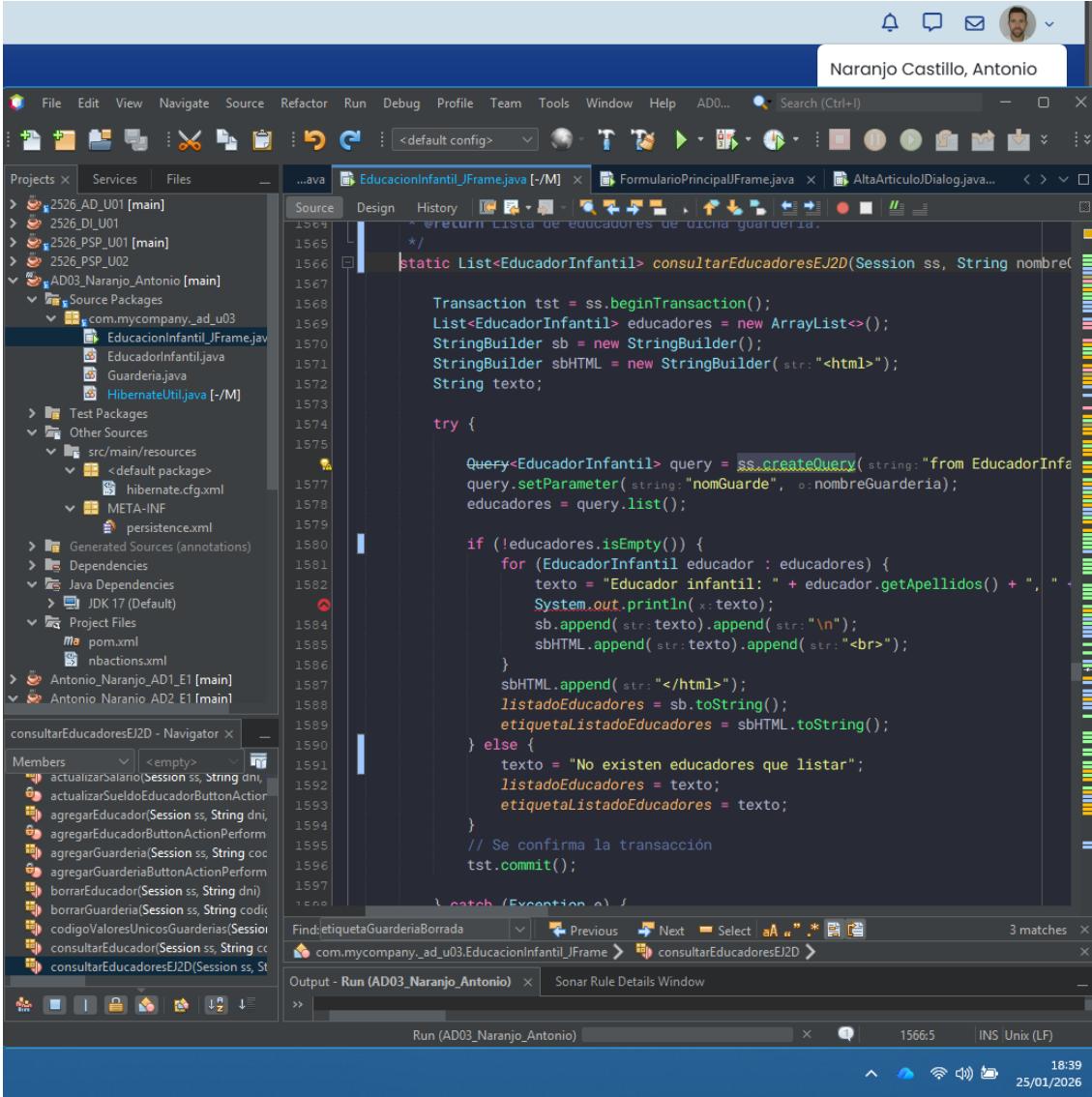
Tras pulsar el botón Listar del apartado Educadores se muestran los educadores que pertenecen a una guardería según selección del usuario por medio del ComboBox. La alimentación del combobox se lleva a cabo previamente realizando una consulta sobre los nombres de las guarderías existentes en la base de datos ejecutando para ello el método refreshComboBoxNombreGuarderias().



The screenshot shows the NetBeans IDE interface with the following details:

- Projects View:** Shows multiple projects including '2526_AD_U01 [main]', '2526_DL_U01', '2526_PSP_U01 [main]', '2526_PSP_U02', 'AD03_Naranjo_Antonio [main]', and 'Antonio Naranjo AD2_E1 [main]'. The 'com.mycompany_ad_u03' package under 'AD03_Naranjo_Antonio' is selected.
- Code Editor:** Displays the Java code for the 'listarEducadoresButtonActionPerformed' method. The code uses Hibernate to query educators from a specific kindergarten. Three yellow arrows point to the following lines of code:
 - // TODO add your handling code here;
 - Session session = HibernateUtil.getSessionFactory().openSession();
 - consultarEducadoresEJ2D(ss, nombreGuarderia);
- Output View:** Shows the output of the run command for 'AD03_Naranjo_Antonio'.
- System Bar:** Shows the date (25/01/2026), time (18:36), and battery level.

Se procede de manera similar a apartados anteriores, pero en este caso se ejecuta el método `consultarEducadoresEJ2D()` recibiendo como argumentos el objeto `Session` y el String nombre de la guardería de la cual se pretende realizar la consulta para recibir una lista de los educadores que forman parte de ella.



```

1564     /**
1565      * @return Lista de educadores de dicha guarderia.
1566     */
1567    static List<EducadorInfantil> consultarEducadoresEJ2D(Session ss, String nombreGuarderia) {
1568
1569        Transaction tst = ss.beginTransaction();
1570        List<EducadorInfantil> educadores = new ArrayList<>();
1571        StringBuilder sb = new StringBuilder();
1572        StringBuilder sbHTML = new StringBuilder( str: "<html>" );
1573        String texto;
1574
1575        try {
1576
1577            Query<EducadorInfantil> query = ss.createQuery( string: "from EducadorInfantil e where e.guarderia.nombre = :nomGuarderia order by e.apellidos asc" );
1578            query.setParameter( string: "nomGuarderia", o:nombreGuarderia );
1579            educadores = query.list();
1580
1581            if (!educadores.isEmpty()) {
1582                for (EducadorInfantil educador : educadores) {
1583                    texto = "Educador infantil: " + educador.getApellidos() + ", " +
1584                        System.out.println( x:texto );
1585                    sb.append( str:texto ).append( str: "\n" );
1586                    sbHTML.append( str:texto ).append( str:<br> );
1587                }
1588                sbHTML.append( str:</html> );
1589                listadoEducadores = sb.toString();
1590                etiquetaListadoEducadores = sbHTML.toString();
1591            } else {
1592                texto = "No existen educadores que listar";
1593                listadoEducadores = texto;
1594                etiquetaListadoEducadores = texto;
1595            }
1596            // Se confirma la transacción
1597            tst.commit();
1598        } catch (Exception e) {
1599        }
1600    }

```

The screenshot shows the NetBeans IDE interface with the code editor open. The code implements a static method named 'consultarEducadoresEJ2D' that takes a Session object and a string representing the name of a kindergarten. It begins by starting a transaction and creating a query to find all educators associated with the specified kindergarten, ordered by their last names. If there are results, it iterates through them, printing each educator's name to the console and appending it to a StringBuilder object. This StringBuilder is then used to build an HTML string for display. If no results are found, it simply returns a message indicating that no educators were found. Finally, it commits the transaction.

Se muestra el método `consultarEducadoresEJ2D()` procediendo de manera similar a los casos anteriores, creando un objeto `Transaction`, una lista `Query` definida por una consulta HQL “`from EducadorInfantil e where codigoGuarderia.nombre = :nomGuarderia order by e.apellidos asc`”, es decir, lista de educadores que pertenecen a la guardería indicada por el usuario (combobox) y ordenadas alfabéticamente por apellido. Por lo demás, se procede como en apartados anteriores.

e) Mostrar un listado con todos los educadores infantiles que ganan más de una determinada cantidad indicada por el usuario (ordenados por salario de mayor a menor).

1) Añadir Objetos

Guarderías

- Código:
- Nombre:
- Capacidad:
- Presupuesto:

Educadores

- DNI: Fecha Nac:
- Nombre: Fecha Alta:
- Apellidos: Salario:
- Código Guardería:

2) Eliminar Objetos

Guarderías

- Código: ABC01

Educadores

- DNI: 11111111A

3) Listar Guarderías

Guardería: Centro Aventura --Código: AE
Guardería: Los arrecifes --Código: ABC01
Guardería: Barcos de Papel --Código: C
Guardería: La casita de Noa --Código: D

4) Listar Educadores

Salario mayor que:

Message

Educador infantil: GOMEZ GOMEZ, LAURA --DNI: 44444444D--Salario: 40000.0
Educador infantil: LUNA PIO, MARCOS --DNI: 88888888J--Salario: 38000.0
Educador infantil: LOPEZ FUENTES, ELISA --DNI: 10101010L--Salario: 35000.0
Educador infantil: LOPEZ MARIN, PEPE --DNI: 11111111A--Salario: 35000.0
Educador infantil: CASAS CAVA, CARMEN --DNI: 77777777H--Salario: 35000.0
Educador infantil: PEREZ PEREZ, LOLA --DNI: 22222222B--Salario: 32000.0

5) Listar Educadores

Salario mayor que:

Educador infantil: GOMEZ GOMEZ, LAURA --DNI: 44444444D--Salario: 40000.0
Educador infantil: LUNA PIO, MARCOS --DNI: 88888888J--Salario: 38000.0
Educador infantil: LOPEZ FUENTES, ELISA --DNI: 10101010L--Salario: 35000.0
Educador infantil: LOPEZ MARIN, PEPE --DNI: 11111111A--Salario: 35000.0
Educador infantil: CASAS CAVA, CARMEN --DNI: 77777777H--Salario: 35000.0
Educador infantil: PEREZ PEREZ, LOLA --DNI: 22222222B--Salario: 32000.0

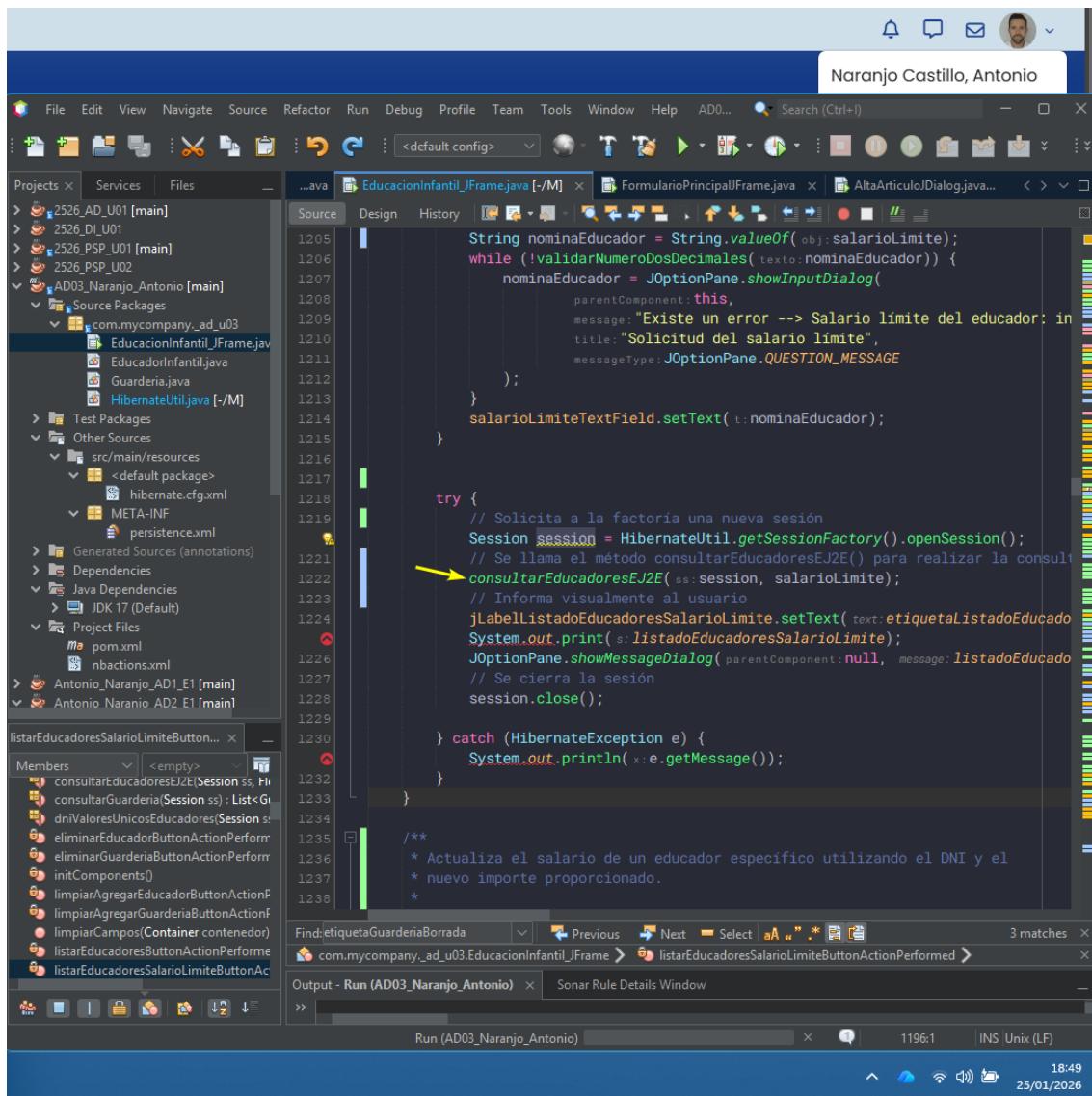
6) Actualizar salario Educador Infantil

DNI: 11111111A

Nombre del educador a eliminar:

18:46 25/01/2026

En este quinto apartado se listan todos los educadores que a modo de ejemplo tengan un sueldo superior a 30.000 €, previa a la validación de los datos introducidos por el usuario en el campo de texto, se procede a ejecutar el código que se encuentra tras pulsar el botón Listar. Se muestran en pantalla como en puntos anteriores el listado de educadores tanto en una etiqueta objeto JLabel como en un objeto JOptionPane.



```

1205     String nominaEducador = String.valueOf(obj:salarioLimite);
1206     while (!validarNumeroDosDecimales(texto:nominaEducador)) {
1207         nominaEducador = JOptionPane.showInputDialog(
1208             parentComponent:this,
1209             message:"Existe un error --> Salario límite del educador: in",
1210             title:"Solicitud del salario límite",
1211             messageType:JOptionPane.QUESTION_MESSAGE
1212         );
1213     }
1214     salarioLimiteTextField.setText(t:nominaEducador);
1215
1216
1217     try {
1218         // Solicita a la factoría una nueva sesión
1219         Session session = HibernateUtil.getSessionFactory().openSession();
1220         // Se llama el método consultarEducadoresEJ2E() para realizar la consulta
1221         consultarEducadoresEJ2E(ss:session, salarioLimite);
1222         // Informa visualmente al usuario
1223         jLabelListadoEducadoresSalarioLimite.setText(text:etiquetaListadoEducado);
1224         System.out.print(s:listadoEducadoresSalarioLimite);
1225         JOptionPane.showMessageDialog(parentComponent:null, message:listadoEducado);
1226         // Se cierra la sesión
1227         session.close();
1228
1229     } catch (HibernateException e) {
1230         System.out.println(e.getMessage());
1231     }
1232
1233     /**
1234      * Actualiza el salario de un educador específico utilizando el DNI y el
1235      * nuevo importe proporcionado.
1236     */
1237
1238

```

The screenshot shows the NetBeans IDE interface with the following details:

- Projects Tab:** Shows several projects including "2526_AD_U01 [main]", "2526_DL_U01", "2526_PSP_U01 [main]", "2526_PSP_U02", "AD03_Naranjo_Antonio [main]", "Source Packages", "Test Packages", "Other Sources", "src/main/resources", "Java Dependencies", "Project Files", "Antonio_Naranjo_AD1_E1 [main]", and "Antonio Naranjo AD2 E1 [main]."
- Code Editor:** Displays the Java code for "EducacionInfantil.JFrame.java". A yellow arrow points to the line `consultarEducadoresEJ2E(ss, salarioLimite);` which is part of a try block.
- Toolbars and Status Bar:** Standard NetBeans toolbars and status bar showing the run configuration ("Run (AD03_Naranjo_Antonio)", "Sonar Rule Details Window"), file count (1196:1), and date/time (18:49, 25/01/2026).

Como punto diferencial de apartados anteriores se ejecuta el método `consultarEducadoresEJ2E()` que recibe como argumento el objeto `Session` y un `Float` `salario límite` a partir del cual se realiza la consulta.

```

1611     * @return Lista de educadores que cumplen la condición.
1612     */
1613     static List<EducadorInfantil> consultarEducadoresEJ2E(Session ss, Float salarioLimite) {
1614         Transaction tst = ss.beginTransaction();
1615         List<EducadorInfantil> educadores = new ArrayList<>();
1616         StringBuilder sb = new StringBuilder();
1617         StringBuilder sbHTML = new StringBuilder(str: "<html>");
1618         String texto;
1619
1620         try {
1621             Query<EducadorInfantil> query = ss.createQuery(string: "from EducadorInfantil e where salario > :sueldoLimite order by e.salario desc");
1622             query.setParameter(string: "sueldoLimite", o: salarioLimite);
1623             educadores = query.list();
1624
1625             if (!educadores.isEmpty()) {
1626                 for (EducadorInfantil educador : educadores) {
1627                     texto = "Educador infantil: " + educador.getApellidos() + ", ";
1628                     System.out.println(x:texto);
1629                     sb.append(str:texto).append(str: "\n");
1630                     sbHTML.append(str:texto).append(str: "<br>");
1631                 }
1632                 sbHTML.append(str:"</html>");
1633                 listadoEducadoresSalarioLimite = sb.toString();
1634                 etiquetaListadoEducadoresSalarioLimite = sbHTML.toString();
1635             } else {
1636                 texto = "No existen educadores que listar";
1637                 listadoEducadoresSalarioLimite = texto;
1638                 etiquetaListadoEducadoresSalarioLimite = texto;
1639             }
1640             // Se confirma la transacción
1641             tst.commit();
1642         } catch (Exception e) {
1643             e.printStackTrace();
1644         }
1645     }

```

Find:etiquetaGuarderiaBorrada Previous Next Select a/* * Output - Run (AD03_Naranjo_Antonio) Sonar Rule Details Window

De la misma manera que en otras ocasiones se devuelve una lista de objetos EducadorInfantil previa consulta HQL “from EducadorInfantil e where salario > :sueldoLimite order by e.salario desc” recogiéndose en una lista Query, si la lista no está vacía se procede a alimentar las variables estáticas de la clase principal que formatearán los resultados a mostrar.

f) Modificar el salario de un educador a una determinada cantidad (el usuario debe facilitar dni del educador y nuevo salario).

The screenshot shows a software interface for managing educators. At the top, there's a header bar with the user's name 'Naranjo Castillo, Antonio'. Below it is a toolbar with icons for notifications, messages, and account settings. The main area is divided into several sections:

- Educadores (Top Left):** A form for adding new educators. It includes fields for DNI, Nombre, Apellidos, Fecha Nacimiento, Fecha Alta, and Salario. Buttons for 'Limpiar' (Clear) and 'Añadir' (Add) are at the bottom.
- Educadores (Top Right):** A list of educators pending addition. It shows the message 'Educador infantil pendiente de añadir'.
- Educadores (Bottom Left):** A list of educators with their details. It includes a 'Borrar' (Delete) button and a 'Listar' (List) button.
- 4) Listar Educadores (Bottom Center):** A list of educators from different centers. It shows entries like 'COSTA RUBIO, REMEDIOS -DNI: 33333333C-Salario: 25000.0', 'LOPEZ MARIN, PEPE -DNI: 11111111A-Salario: 35000.0', and 'PEREZ PEREZ, LOLA -DNI: 22222222B-Salario: 32000.0'.
- 6) Actualizar salario Educador Infantil (Bottom Right):** A dialog box for updating an educator's salary. It has fields for 'DNI' (set to '44444444D') and 'Nuevo Salario' (set to '42560.7'). An 'Actualizar' (Update) button is at the bottom right. A yellow arrow points to the 'Actualizar' button, and another points to the success message below it: 'El sueldo del educador LAURA GOMEZ GOMEZ se ha actualizado con éxito'.

En este último apartado se procede a modificar o actualizar el salario de un educador seleccionado entre los presentes, se elige el educador con DNI 44444444D que anteriormente tenía un salario de 40000.0 € y se actualiza a 42560.7 €. Se muestra una etiqueta con el resultado obtenido tras llevar a cabo la actualización.

Naranjo Castillo, Antonio

```

git-cmd - .\mysqld
Símbolo del sistema - .\mysql
+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 40000 | ABC02
| 5555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03
| 9999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04
+-----+
10 rows in set (0.011 sec)

mysql> SELECT * FROM guarderia;
+-----+
| codigo | nombre | capacidad | presupuesto |
+-----+
| ABC01 | Centro Aventura | 50 | 20000
| ABC02 | Los arrecifes | 78 | 45000
| CDE03 | Barcos de Papel | 38 | 15000
| DFG04 | La casita de Noa | 48 | 40000
+-----+
4 rows in set (0.012 sec)

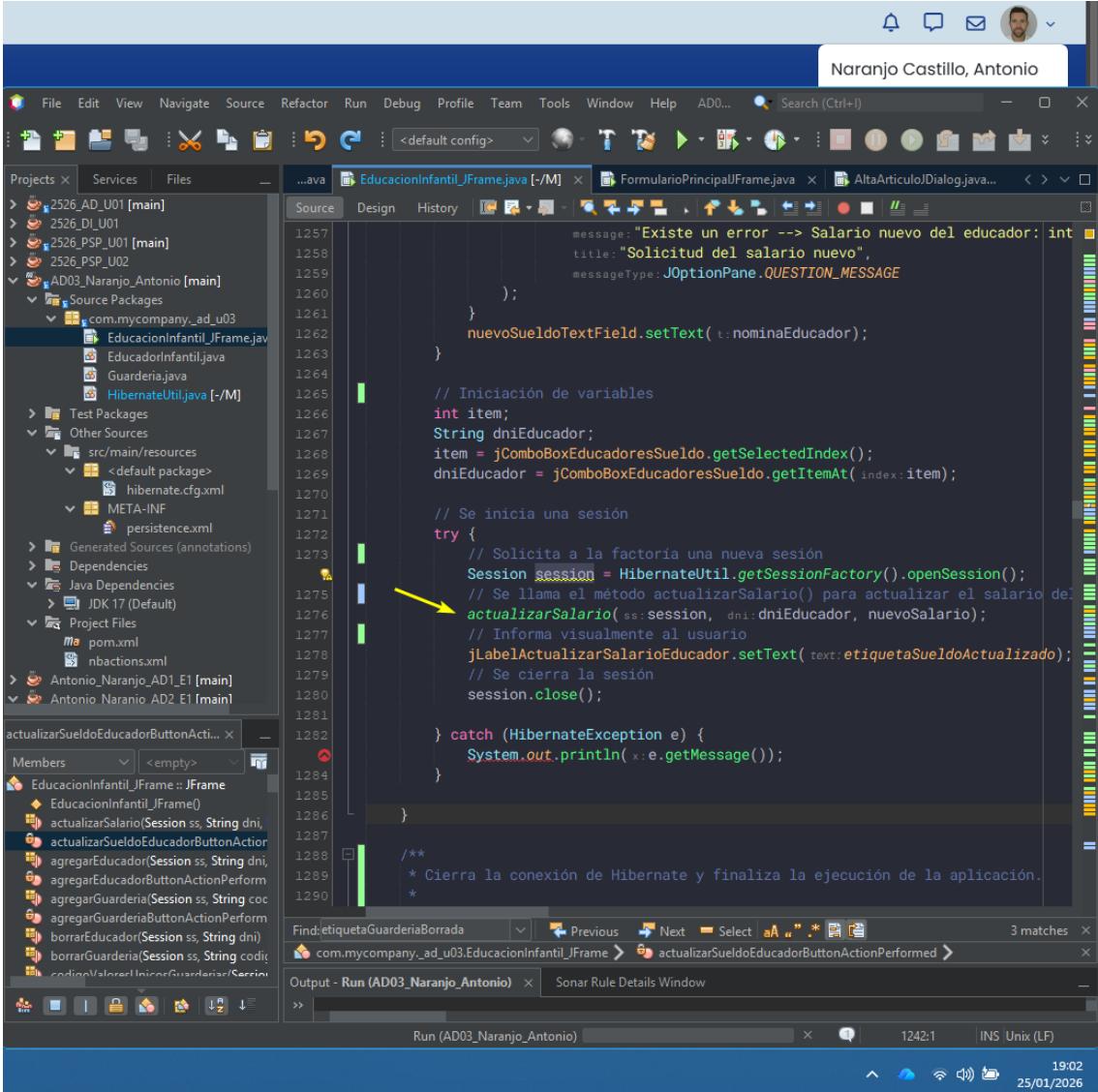
mysql> SELECT * FROM educador_infantil;
+-----+
| dni | nombre | apellidos | fecha_nacimiento | fecha_alta | salario | codigo_guarderia |
+-----+
| 10101010L | ELISA | LOPEZ FUENTES | 1990-03-02 | 2015-07-04 | 35000 | DFG04
| 11111111A | PEPE | LOPEZ MARIN | 1984-03-02 | 2010-03-01 | 35000 | ABC01
| 22222222B | LOLA | PEREZ PEREZ | 1988-01-01 | 2010-03-01 | 32000 | ABC01
| 33333333C | REMEDIOS | COSTA RUBIO | 1996-03-03 | 2015-01-01 | 25000 | ABC01
| 44444444D | LAURA | GOMEZ GOMEZ | 1978-01-20 | 2010-08-01 | 42560.7 | ABC02
| 5555555F | JUAN | LOPEZ LOPEZ | 1988-05-01 | 2008-05-03 | 25000 | ABC02
| 66666666G | ANA | MARIN MARIN | 2000-03-02 | 2020-09-01 | 18000 | ABC02
| 77777777H | CARMEN | CASAS CAVA | 1975-09-25 | 2000-05-08 | 35000 | CDE03
| 88888888J | MARCOS | LUNA PIO | 1989-08-15 | 2010-01-01 | 38000 | CDE03
| 9999999K | JUANA | NAVARRO MARIN | 1998-04-05 | 2015-04-06 | 29500 | DFG04
+-----+
10 rows in set (0.022 sec)

mysql> |

```

19:00
25/01/2026

Se consulta en el cliente MySQL que la tabla educador_infantil se han actualizado correctamente los datos del educador seleccionado reflejando el salario anteriormente introducido.



The screenshot shows the NetBeans IDE interface with the following details:

- Projects Tab:** Shows several projects including "2526_AD_U01 [main]", "2526_DL_U01", "2526_PSP_U01 [main]", "2526_PSP_U02", "AD03_Naranjo_Antonio [main]", and "Antonio_Naranjo_E1 [main]".
- Source Packages:** Under "AD03_Naranjo_Antonio [main] / Source Packages / com.mycompany_ad_u03", the "EducacionInfantil_JFrame.java" file is open.
- Code Editor:** The code for the `actualizarSueldoEducadorButtonActionPerformed` method is displayed. A yellow arrow points to the line `actualizarSalario(ss, dniEducador, nuevoSalario);`.
- Output Tab:** Shows the output for the run "Run (AD03_Naranjo_Antonio)".
- Bottom Status Bar:** Displays the date "25/01/2026" and time "19:02".

```

1257     message:"Existe un error --> Salario nuevo del educador: int
1258     title:"Solicitud del salario nuevo",
1259     messageType: JOptionPane.QUESTION_MESSAGE
1260   );
1261 }
1262 nuevoSueldoTextField.setText( t:nominaEducador );
1263 }
1264
1265 // Iniciación de variables
1266 int item;
1267 String dniEducador;
1268 item = jComboBoxEducadoresSueldo.getSelectedIndex();
1269 dniEducador = jComboBoxEducadoresSueldo.getItemAt( index:item );
1270
1271 // Se inicia una sesión
1272 try {
1273   // Sigue la ejecución de la aplicación
1274   Session session = HibernateUtil.getSessionFactory().openSession();
1275   // Se llama el método actualizarSalario() para actualizar el salario de
1276   actualizarSalario( ss:session, dni:dniEducador, nuevoSalario );
1277   // Informa visualmente al usuario
1278   jLabelActualizarSalarioEducador.setText( text:etiquetaSueldoActualizado );
1279   // Se cierra la sesión
1280   session.close();
1281
1282 } catch (HibernateException e) {
1283   System.out.println( x:e.getMessage() );
1284 }
1285
1286 }
1287 /**
1288 * Cierra la conexión de Hibernate y finaliza la ejecución de la aplicación.
1289 */
1290

```

Como en otras ocasiones se procede de manera similar, pero ejecutando de manera diferencial el método `actualizarSalario()` que recibe como argumentos el objeto `Session`, el String DNI del educador y el Float salario del educador.

```

1657     /*
1658      *param nuevoSalario
1659      */
1660      static void actualizarSalario(Session ss, String dni, Float nuevoSalario) {
1661          Transaction tst = ss.beginTransaction();
1662
1663          try {
1664              EducadorInfantil educador = ss.get( type:EducadorInfantil.class, srlbl:DNI );
1665
1666              if (educador == null) {
1667                  System.out.println("No se encontró el educador con DNI: " + dni);
1668              } else {
1669                  // Actualizar guardería
1670                  educador.setSalario( salario:nuevoSalario );
1671                  ss.update( e:educador );
1672                  System.out.println("Se actualizó el sueldo del educador con DNI: " + dni );
1673                  etiquetaSueldoActualizado = "El sueldo del educador " + educador.get
1674
1675                  // Se confirma la transacción
1676                  tst.commit();
1677
1678              }
1679
1680          } catch (Exception e) {
1681
1682              if (tst != null) {
1683                  // Hacer rollback en caso de error
1684                  tst.rollback();
1685              }
1686              System.out.println( x:e.getMessage() );
1687
1688          }
1689
1690      /**
1691      * Obtiene una lista de los DNI's de todos los educadores sin duplicados
1692      */
1693  }

```

The screenshot shows the IntelliJ IDEA interface with the code editor open. The code is written in Java and performs the following steps:

- Creates a new Transaction object.
- Attempts to find an EducadorInfantil by its DNI.
- If found, updates the salary and persists the changes.
- Commits the transaction to apply the changes.
- Catches any exceptions and rolls back the transaction if one occurs.
- Prints the error message to the console.

En el método `actualizarSalario()` se crea un objeto `Transaction`, se crea un objeto `EducadorInfantil` con el mismo DNI que el aportado como argumento, se comprueba que no sea un elemento nulo, sino que existe en la base de datos, se establece el nuevo salario pasado como argumento, se ejecuta el método `update()` del objeto `Session` para dar persistencia a los nuevos datos aportados, para luego mediante el método `commit()` del objeto `Transaction` aplicar los cambios. Si existe algún error serán captado y se ejecutará el método `rollback()` para deshacer los cambios.