

Técnico Superior en Desarrollo de Aplicaciones Multimedia

Curso Académico 2021/2022 - 2º DAMS

Alumno: José Enrique Jordán Moreno

Unidad 3: Mapeo objeto-relacional usando Hibernate

Contenido

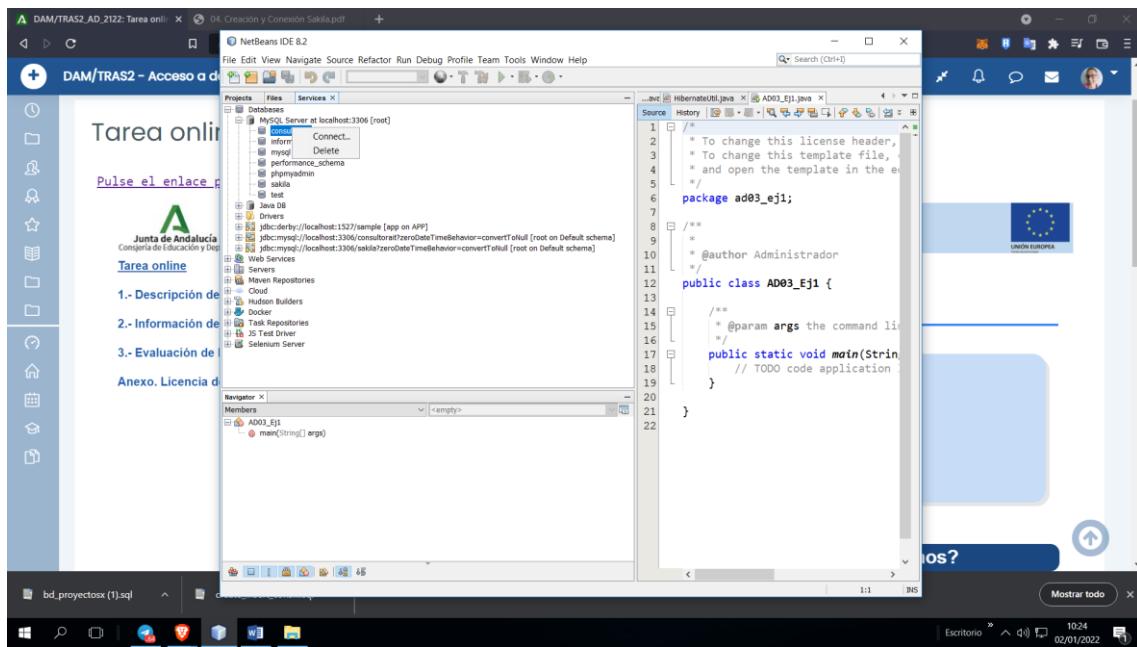
1. Ejercicio 1	4
Desarrolla un proyecto en Netbeans y configura Hibernate para realizar el mapeo de esta base de datos relacional.....	4
1.1.1 Crea el fichero de configuración de Hibernate (hibernate.cfg.xml), con conexión a la base de datos consultorait con JDBC.	4
1.1.2 .Crea el fichero de ingeniería inversa hibernate.reveng.xml e indica las tablas sobre las que se va a establecer correspondencia.....	8
1.1.3. Genera las clases y los ficheros de correspondencia (hbm).	9
1.1.4. Codifica y genera las siguientes consultas mediante HQL	11
Consulta 1: Apellido, nombre, salario y número de empleado con un salario inferior a 1600.....	11
Consulta 2: Número de empleado, departamento y salario de los empleados, ordenados de menor a mayor salario.	12
Consulta 3: Datos de empleados cuyo departamento no esté en GRANADA.....	12
Consulta 4: Apellido, salario y número de departamento de los empleados cuyo salario sea mayor que el máximo salario del departamento 10.....	13
Consulta 5: Empleados con salario menor que alguno de los empleados del departamento 20.....	14
Consulta 6: Mostrar nombre y total de empleados de aquellos departamentos con más de un empleado adscrito. Ordena el resultado por número de empleado.....	14
2. Ejercicio 2	15
Crea un proyecto Netbean llamado AD03_Ej2_ApellidosNombre que realice un mapeo objeto-relacional con Hibernate sobre la siguiente base de datos:.....	15
2.1. Una vez creado el proyecto y realizado el mapeo de los objetos de la BD (incluidas las relaciones), observa los ficheros de mapeo generados y las correspondientes clases Java (POJO):	15
2.1.1 Realiza un estudio sobre las clases generadas, interpretando y describiendo brevemente el mapeo realizado por Hibernate. Acompaña tus explicaciones de capturas de pantalla.	16
2.1.2 ¿Cambia algo en el fichero de configuración de Hibernate?.....	17
2.1.3 ¿Qué ocurre con los nombres de los atributos de las clases (por ejemplo: nom_dept) ?	18
2.1.4 ¿Se llaman igual que los atributos de las tablas?	18
2.2 Crea y añade a tu proyecto los siguientes métodos:	19

2.2.1 Inserta un nuevo objeto Empleados en la BD (código empleado: 'A22', nombre: 'Pepe García', fecha de ingreso: 'la fecha actual', salario: 2000, jefe: 'A11' y departamento '01'). Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha insertado correctamente.	19
2.2.2 Modifica el salario del objeto Empleados de la BD creado en el apartado anterior (empleado A22) con un nuevo valor de 2500. Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha modificado correctamente.....	20
2.2.3 Elimina un objeto Empleados de la BD (por ejemplo el empleado A22). Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha eliminado correctamente.....	22
2.2.4 Realiza (codifica) la siguiente consulta: "Empleados con salario superior a 1000 € (mostrar su nombre, número de departamento y salario)".	22

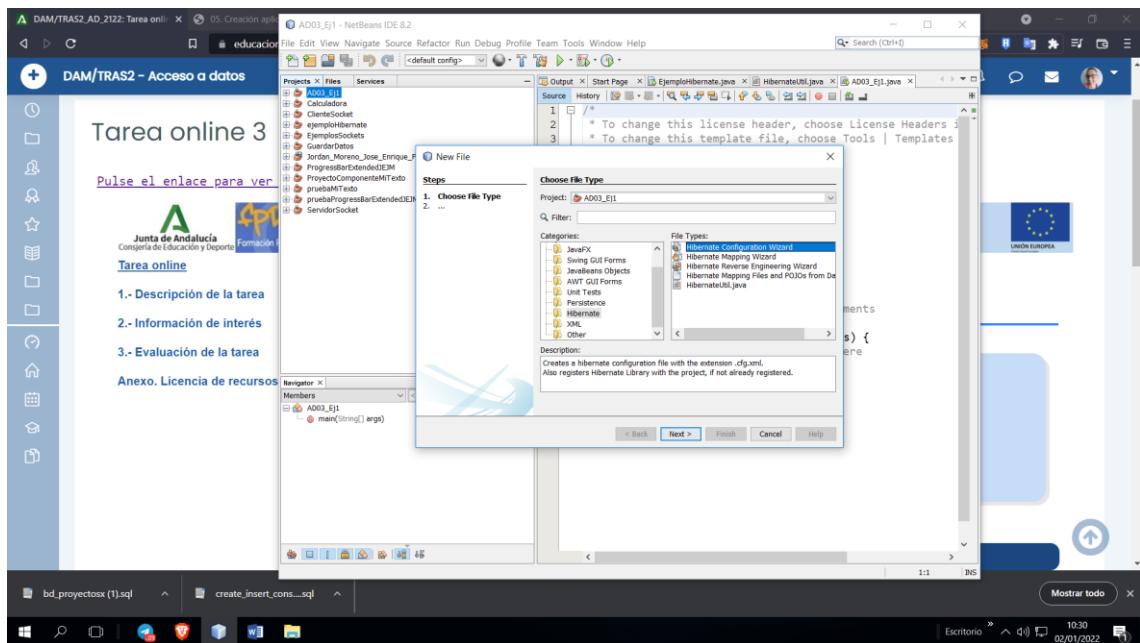
1. Ejercicio 1.

Desarrolla un proyecto en Netbeans y configura Hibernate para realizar el mapeo de esta base de datos relacional.

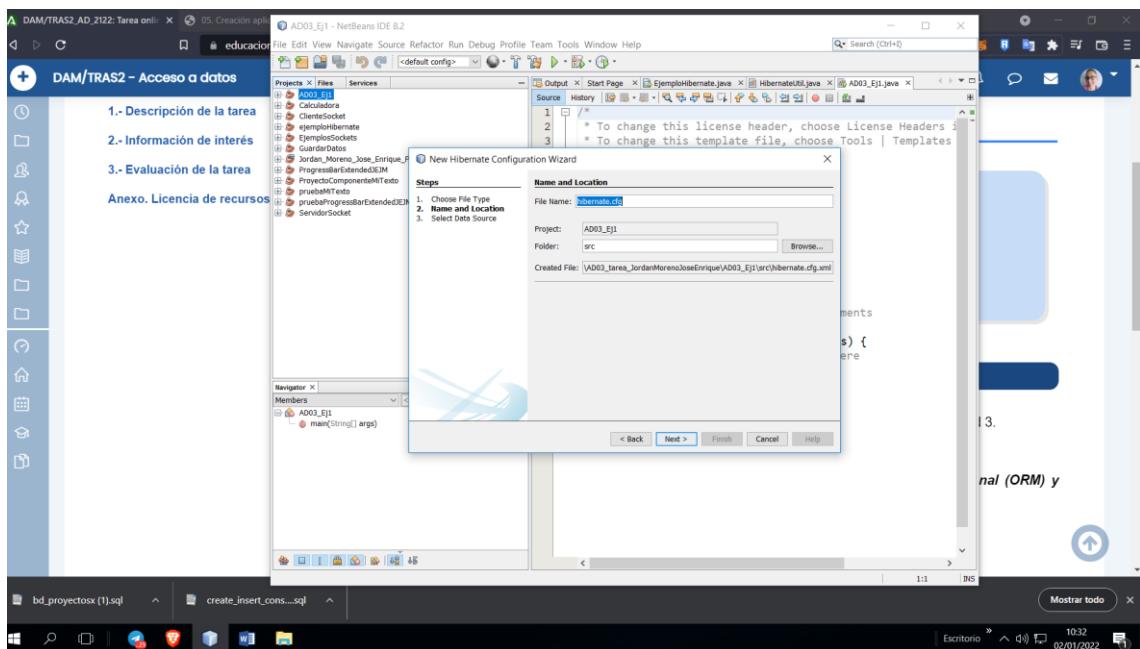
1.1.1 Crea el fichero de configuración de Hibernate (hibernate.cfg.xml), con conexión a la base de datos consultorait con JDBC.



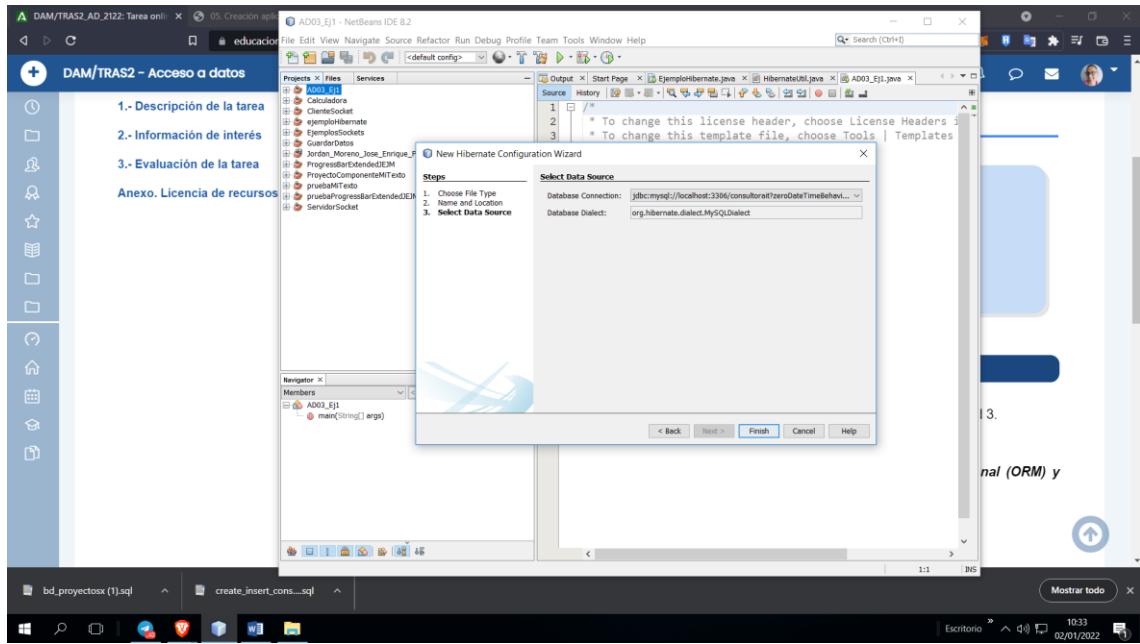
Para crear el fichero de configuración de Hibernate en Netbeans primero en Services Databases con el ratón seleccionamos connect y conectamos con la bbdd .



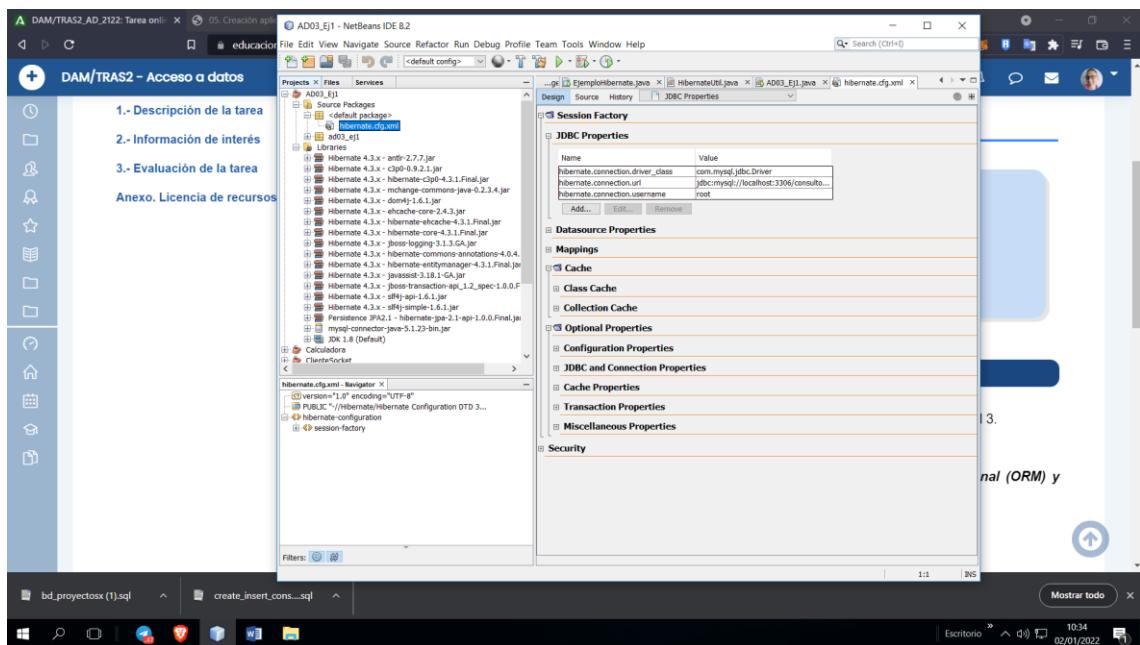
Seleccionamos el proyecto y sobre el proyecto añadimos Hibernate con ayuda del asistente.



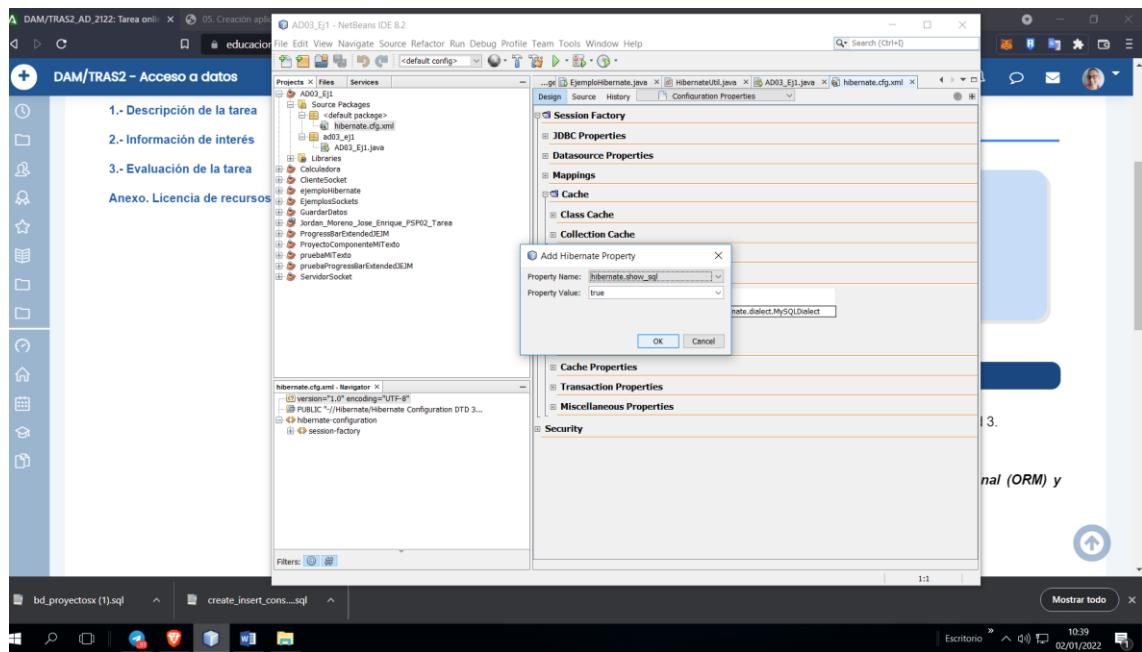
De ésta forma creamos el fichero hibernate.cfg , desde el asistente.



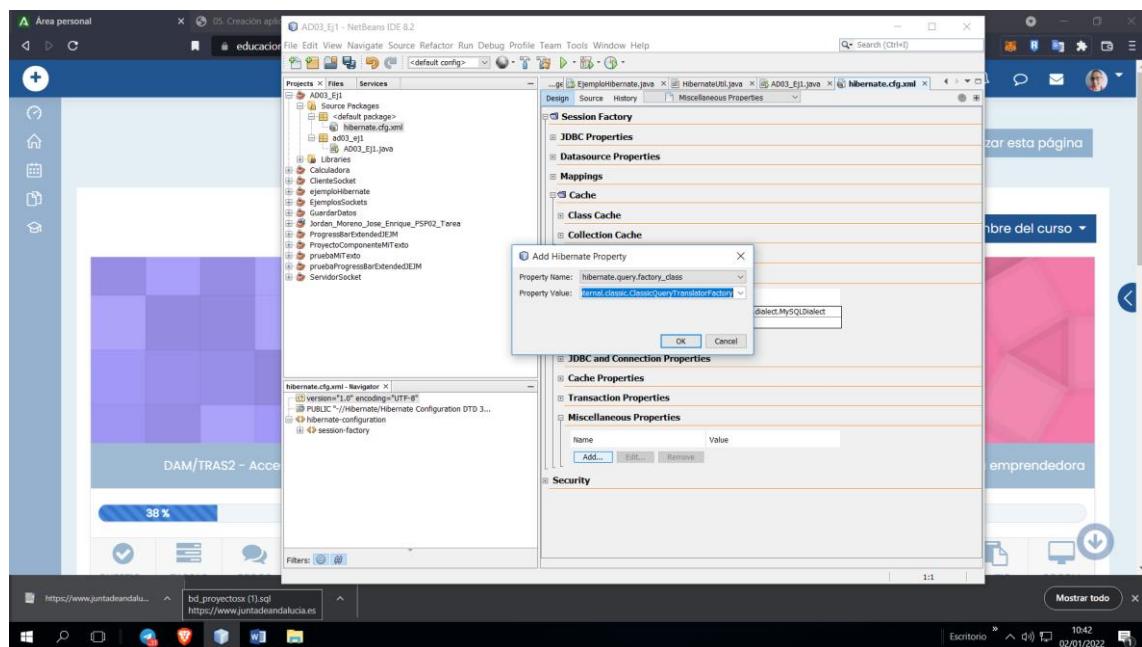
Seleccionamos la bbdd consultarit.



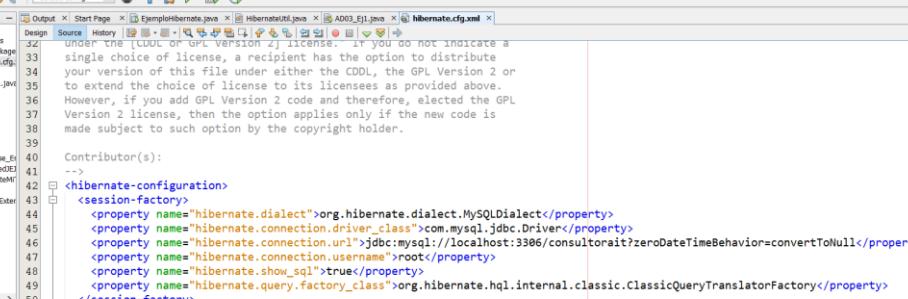
Comprobamos que se han incorporado al proyecto las librerías de Hibernate en Libraries y abrimos el fichero **hibernate.cfg.xml**



Añadimos la propiedad **hibernate.show.sql**



**Hibernate.query.factory_class,
org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory**



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

- Title Bar:** AD03_Ej1 - NetBeans IDE 8.2
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Search Bar:** Search (Ctrl+F)
- Code Editor:** The main window displays the `hibernate.cfg.xml` file. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by MyEclipse Persistence Tools. -->
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/consultorait?zeroDateTimeBehavior=convertToNull</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.query.factory_class">org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory</property>
    </session-factory>
</hibernate-configuration>
```

- Sidebar:** Shows the project structure with files like `ejemploHibernate.java`, `AD03_S1.java`, and `hibernate.cfg.xml`.
- Bottom:** Shows the database connection configuration (`bd_proyectosos1.sql`) and a SQL query (`create_insert_cons...sql`).

Configuración del fichero hibernate.cfg.xml

1.1.2 .Crea el fichero de ingeniería inversa hibernate.reveng.xml e indica las tablas sobre las que se va a establecer correspondencia.

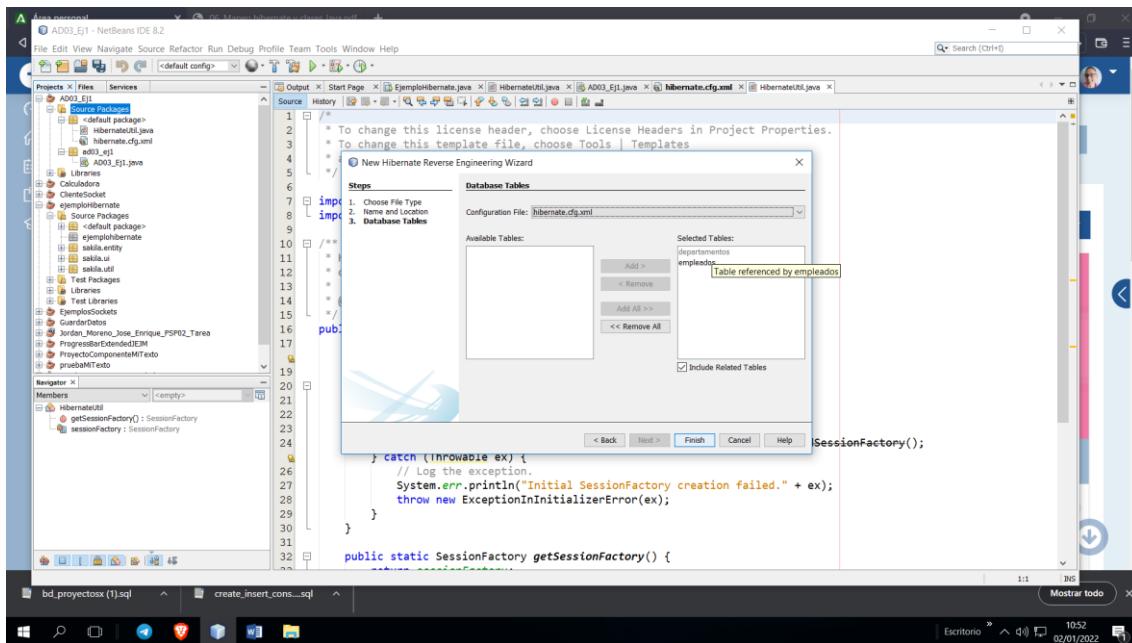
The screenshot shows the NetBeans IDE interface with the title bar "AD03_Ej1 - NetBeans IDE 8.2". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for New, Open, Save, Cut, Copy, Paste, Find, Replace, Select All, Undo, Redo, and others. The main window displays a Java code editor with the following code:

```
private static final SessionFactory sessionFactory;
static {
    try {
        // Create the SessionFactory from standard (hibernate.cfg.xml)
        // config file.
        sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
    } catch (Throwable ex) {
        // Log the exception.
        System.err.println("Initial SessionFactory creation failed." + ex);
        throw new ExceptionInInitializerError(ex);
    }
}
public static SessionFactory getSessionFactory() {
```

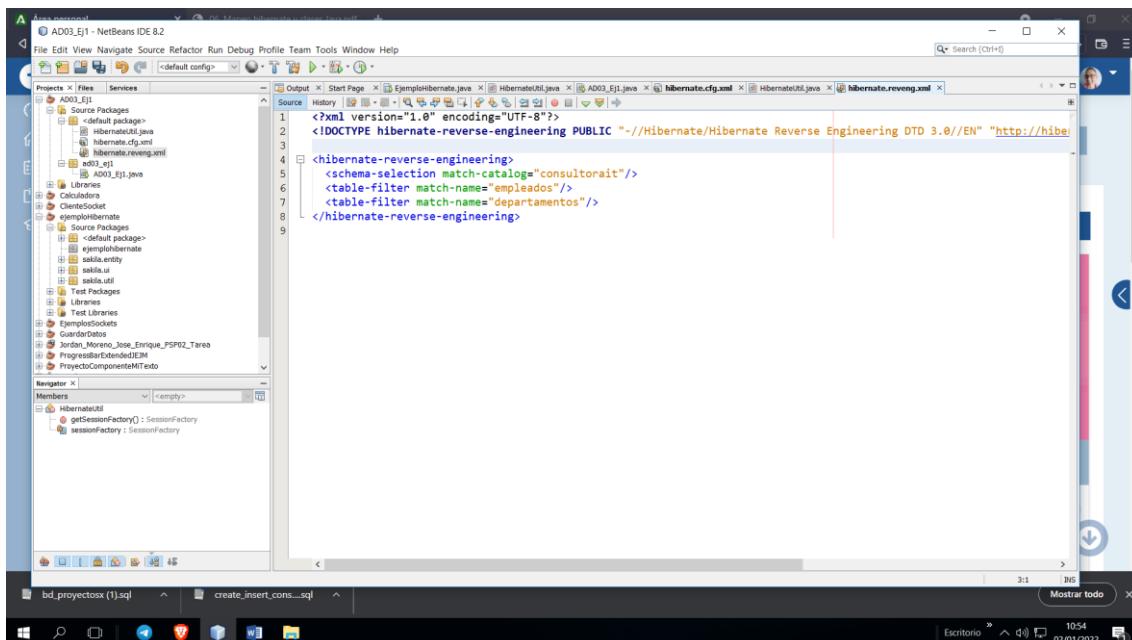
A context menu is open over the line "sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();". The menu path "New > Reverse Engineering Wizard..." is highlighted. Other options in the menu include Folder..., HibernateUtil.java..., Hibernate Configuration Wizard..., JFrame Form..., Hibernate Mapping Files and POJOs from Database..., Java Class..., Java Package..., Property Editor..., JPanel Form..., JavaBeans Component..., Class Diagram..., Empty File..., HTML File..., Java Interface..., Entity Class..., and Other... . The status bar at the bottom right shows "11:1 INS Mostrar todo".

Para crear el fichero **hiberante.reveng.xml**. En el proyecto con el botón derecho pulsamos en el Asistente de Hibernate para crear el fichero de ingeniería inversa.

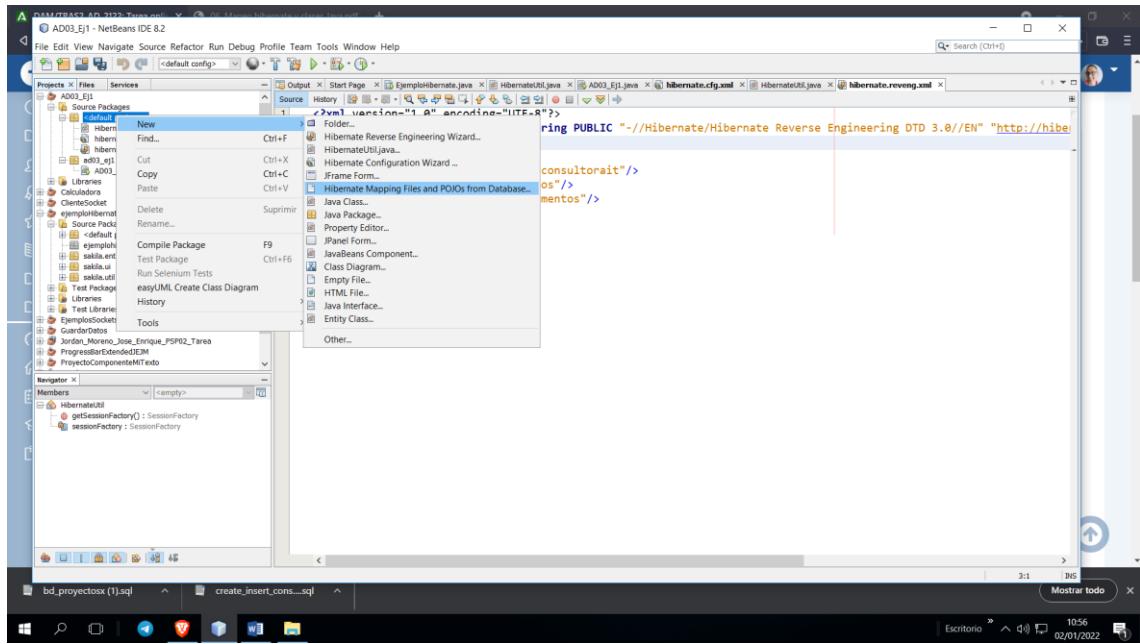
1.1.3. Genera las clases y los ficheros de correspondencia (hbm).



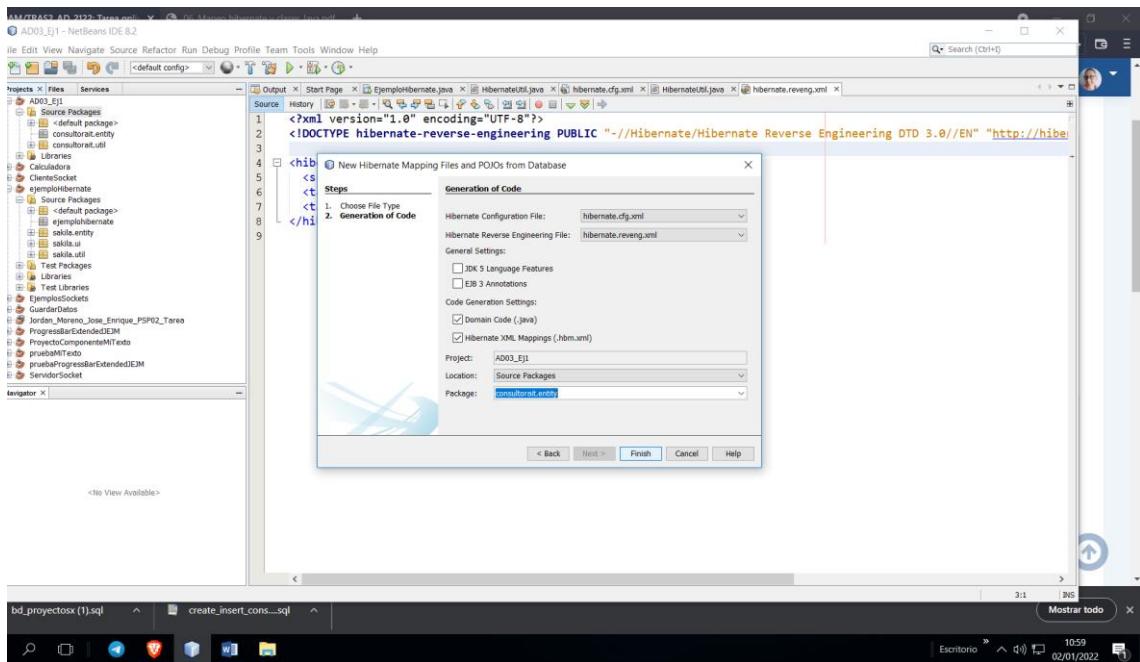
Seleccionamos las tablas con las que vamos a trabajar, en este caso departamentos y empleados.



Podemos comprobar que en el fichero hibernate.reveng.xml se han creado varios table-filter para las tablas empleados y departamentos.



Ahora abrimos el asistente de Netbeans para generar los ficheros POJO de la BBDD.



Guardamos los ficheros POJO en un nuevo paquete **consultarit.entity**

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

- Title Bar:** AD03_E1 - NetBeans IDE 8.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Source Editor:** Displays the `Departamentos.hbm.xml` file content. The code is as follows:

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 02-ene-2022 11:00:15 by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
  <class name="consultorait.entity.Departamentos" table="departamentos" catalog="consultorait" optimistic-lock="true">
    <id name="deptNo" type="byte">
      <column name="dept_no" />
      <generator class="assigned" />
    </id>
    <property name="dnombre" type="string">
      <column name="dnombre" length="20" />
    </property>
    <property name="loc" type="string">
      <column name="loc" length="15" />
    </property>
    <set name="empleados" table="empleados" inverse="true" lazy="true" fetch="select">
      <key>
        <column name="dept_no" not-null="true" />
      </key>
      <one-to-many class="consultorait.entity.Empleados" />
    </set>
  </class>
</hibernate-mapping>

```

- Project Explorer:** Shows the project structure with packages like `AD03_E1`, `Source Packages`, and `Libraries`.
- Navigator:** Shows the `Departamentos.hbm.xml` file selected.
- Bottom Status Bar:** Shows the date and time (02/01/2022, 11:00) and the message "Mostrar todo".

Comprobamos que se han creado los ficheros Departamentos y Empleados.hbm.xml y sus ficheros .java correspondientes.

1.1.4. Codifica y genera las siguientes consultas mediante HQL

Consulta 1: Apellido, nombre, salario y número de empleado con un salario inferior a 1600.

```
SELECT E.apellido, E.nombre, E.empNo FROM Empleados E WHERE E.salario > 1600
```

The screenshot shows the NetBeans IDE interface with the title bar "DAM/TRAS2_AD_2122: Dudas sol..." and "HQL Insert, HQL Insert Query Example Tarea Online 3.pdf". The main window displays a Java project named "AD03_E1" containing classes like DialogosDatos.java, Empleado.java, Departamentos.java, etc. A SQL query window titled "Session: hibernate.cfg" shows the following HQL:

```
SELECT E.apellido, E.dirección, E.apellido FROM Empleados E
WHERE E.apellido > '1605'
```

The results table shows four rows of data:

Column 1	Column 2	Column 3
CASINELLO	1750.0	2000
DE FRUTOS	3000.0	6000
GONZALEZ	1900.0	7000
ELIZALDE	1850.0	8000

The output window below the query results shows "RUN: BUILD SUCCESSFUL (total time: 2 seconds)".

Consulta 2: Número de empleado, departamento y salario de los empleados, ordenados de menor a mayor salario.

```
SELECT E.empNo, E.departamentos.deptNo, E.salario FROM Empleados E
ORDER by E.salario ASC
```

The screenshot shows the NetBeans IDE interface with the title bar "DAM/TRAS2_AD_2122: Dudas sol..." and "HQL Insert, HQL Insert Query Example Tarea Online 3.pdf". The main window displays a Java project named "AD03_E1" containing classes like DialogosDatos.java, Empleado.java, Departamentos.java, etc. A SQL query window titled "Session: hibernate.cfg" shows the following HQL:

```
SELECT E.apellido, E.departamentos.deptNo, E.salario FROM Empleados E
WHERE E.apellido IN (select d.deptNo from Departamentos d where d.loc != 'GRANADA')
```

The results table shows eight rows of data:

Column 1	Column 2	Column 3
5000	30	1450.0
2000	20	1440.0
1000	20	1250.0
1000	20	1200.0
6000	10	1800.0
6000	40	3000.0
5000	30	1450.0
2000	20	1440.0

The output window below the query results shows "RUN: BUILD SUCCESSFUL (total time: 2 seconds)".

Consulta 3: Datos de empleados cuyo departamento no esté en GRANADA.

```
from Empleados e where e.departamentos IN (select d.deptNo from Departamentos d where d.loc != 'GRANADA')
```

The screenshot shows the NetBeans IDE interface with the title bar "DAM/TRAS2_AD_2122: Dudas sol." and "HQL Insert, HQL Insert Query Example" and "Tarea Online 3.pdf". The main window displays a Java project named "AD03_EJ1 - NetBeans IDE 8.2". In the center, there is a SQL editor window titled "Session: hibernate.cfg" containing the following HQL query:

```
from Empleados e where e.departamentos > (select d.dirigido from Departamento d where d.id=10)
```

Below the query, the "Result: SQL" pane shows the output:

Dir	Cumision	Emplio	FechaAlt	Depart...	Apellido	Oficio	Salario
2	0.0	2900	2019-09-10	consultora...	CASINELLO	ADMINISTRIS.	1750.0
5	5.0	5000	2029-09-31	consultora...	FERMANDEZ	TESTER	1450.0
6	20.0	6000	2017-12-31	consultora...	DE PRUITO	GERENTE	1900.0
8	10.0	7000	2018-03-01	consultora...	GONZALEZ	COMERCIAL	1900.0
9	0.0	8000	2018-01-03	consultora...	BELZUNCES	ADMINSEG.	1850.0

At the bottom of the SQL editor, it says "0 row(s) updated.: 5 row(s) selected.". Below the SQL editor is an "Output" pane showing "BUILD SUCCESSFUL (total time: 2 seconds)".

Consulta 4: Apellido, salario y número de departamento de los empleados cuyo salario sea mayor que el máximo salario del departamento 10.

from Empleados e where e.salario > (select max(e.salario) from Empleados e where e.departamentos= 10)

The screenshot shows the NetBeans IDE interface with the title bar "DAM/TRAS2_AD_2122: Dudas sol." and "HQL Insert, HQL Insert Query Example" and "Tarea Online 3.pdf". The main window displays a Java project named "AD03_EJ1 - NetBeans IDE 8.2". In the center, there is a SQL editor window titled "Session: hibernate.cfg" containing the following HQL query:

```
from Empleados e where e.salario > (select max(e.salario) from Empleados e where e.departamentos= 10)
```

Below the query, the "Result: SQL" pane shows the output:

Apellido	Salario	Departamento No
DE PRUITO	3600.0	10
GONZALEZ	1900.0	10

At the bottom of the SQL editor, it says "0 row(s) updated.: 2 row(s) selected.". Below the SQL editor is an "Output" pane showing "BUILD SUCCESSFUL (total time: 2 seconds)".

Consulta 5: Empleados con salario menor que alguno de los empleados del departamento 20.

```
from Empleados e where e.salario < any (select e.salario from Empleados e  
where e.departamentos = 20)
```

The screenshot shows the NetBeans IDE interface with several tabs open: 'AD03_E1 - NetBeans IDE 8.2', 'File', 'Edit', 'View', 'Navigate', 'Source Refactor', 'Run', 'Debug', 'Profile', 'Team', 'Tools', 'Window', 'Help'. The 'HERRAMIENTAS DE IMAGEN' toolbar is visible at the top. The 'Projects' panel shows a project named 'AD03_E1' with files like 'Utilidades.java', 'Dialogos.java', 'Empleado.java', 'HQL Query0', 'HQL Query1', and 'HQL Query2'. The 'Session' dropdown is set to 'hibernate.cfg'. The 'Result SQL' pane displays the query: 'from Empleados e where e.salario < any (select e.salario from Empleados e where e.departamentos = 20)'. Below it is a table with three rows:

Dir	Comision	EmplNo	FechaAlt	Depart...	Apellido	Oficio	Salario
3	10.0	3000	2019-06-09	consultora...	LOPEZ	DES.WEB	1480.0
4	10.0	4000	2020-09-15	consultora...	GARCIA	DES.MOV.	1500.0
5	5.5	5000	2020-09-31	consultora...	FERNANDEZ	TESTER	1450.0

The 'Output' pane shows the build log: 'RUN: BUILD SUCCESSFUL (total time: 2 seconds)'. The status bar at the bottom right indicates 'Escritorio' at 11:47 on 09/01/2022.

Consulta 6: Mostrar nombre y total de empleados de aquellos departamentos con más de un empleado adscrito. Ordena el resultado por número de empleado.

```
from Empleados e group by e.departamentos having count(*) > 1 order by  
e.empNo
```

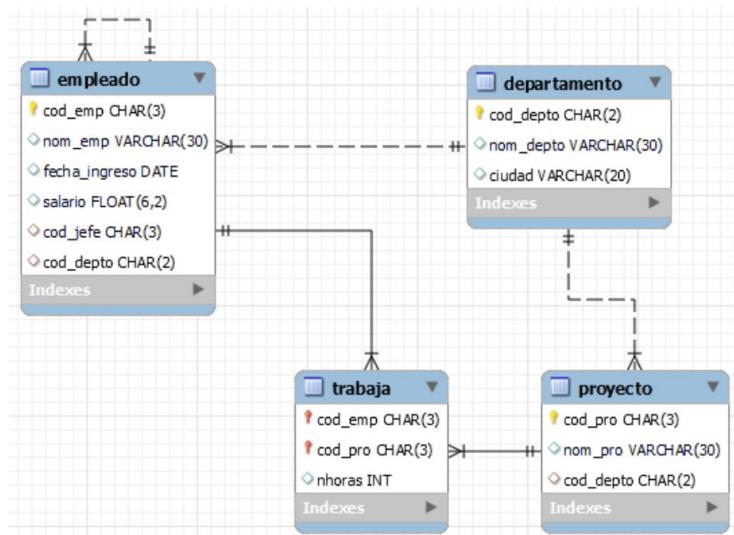
The screenshot shows the NetBeans IDE interface with several windows open. In the center, a 'Session: hibernate.cfg' window displays a SQL query: 'from Empleados e group by e.departamento having count(*) > 1 order by e.apellido'. Below it, the 'Result SQL' window shows a table with two rows:

Dr	Codigo	Apellido	Nombre	FechaIngreso	Departamento	Oficina	Salario
16.0	1000	1000	1000	2010-02-10	Administradora	CASHEIRO	1600.0
16.0	2000	2000	2000	2010-02-10	Administradora	ADMINISTRADOR	1600.0

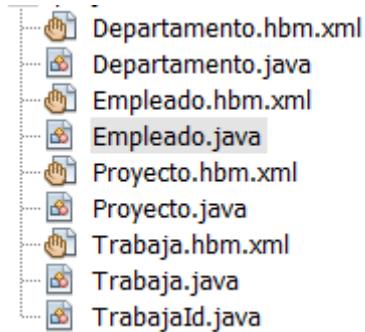
At the bottom right, the 'Output - AD03_Ej2 (run) x' window shows the message: 'BUILD SUCCESSFUL (total time: 2 seconds)'.

2. Ejercicio 2

Crea un proyecto Netbean llamado AD03_Ej2_ApellidosNombre que realice un mapeo objeto-relacional con Hibernate sobre la siguiente base de datos:



2.1. Una vez creado el proyecto y realizado el mapeo de los objetos de la BD (incluidas las relaciones), observa los ficheros de mapeo generados y las correspondientes clases Java (POJO):



Hibernate utiliza los ficheros creados para relacionar las tablas de la bbdd con objetos Java, estos ficheros están en formato XML y que tienen extensión **.hbm.xml**.

Ésta técnica es denominada “**ingeniería inversa**”, se usa para crear archivos de mapeo basados en tablas de la base de datos.

El archivo que se crea es **Hibernate.reveng.xml**, éste archivo se utiliza para modificar la configuración predeterminada de **Hibernate.cfg.xml** y para especificar explícitamente el esquema de base de datos que se va a utilizar, filtrar las tablas que no deseamos recuperar y especificar cómo se asignan los tipos JDBC a los tipos de Hibernate.

2.1.1 Realiza un estudio sobre las clases generadas, interpretando y describiendo brevemente el mapeo realizado por Hibernate. Acompaña tus explicaciones de capturas de pantalla.

```

package projectx.entidad;
// Generated 03-ene-2022 14:04:21 by Hibernate Tools 4.3.1

/**
 * Trabajaid generated by hbm2java
 */
public class Trabajaid implements java.io.Serializable {

    private String codEmp;
    private String codPro;

    public Trabajaid() {
    }

    public Trabajaid(String codEmp, String codPro) {
    }

    public String getCodEmp() {
        return codEmp;
    }

    public void setCodEmp(String codEmp) {
        this.codEmp = codEmp;
    }

    public String getCodPro() {
        return codPro;
    }

    public void setCodPro(String codPro) {
        this.codPro = codPro;
    }

    @Override
    public boolean equals(Object other) {
        if (this == other) {
            return true;
        }
        if (other == null) {
            return false;
        }
        if (getClass() != other.getClass()) {
            return false;
        }
        Trabajaid otherTrabajaid = (Trabajaid) other;
        if (codEmp == null) {
            if (otherTrabajaid.codEmp != null) {
                return false;
            }
        } else if (!codEmp.equals(otherTrabajaid.codEmp)) {
            return false;
        }
        if (codPro == null) {
            if (otherTrabajaid.codPro != null) {
                return false;
            }
        } else if (!codPro.equals(otherTrabajaid.codPro)) {
            return false;
        }
        return true;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += codEmp.hashCode();
        hash += codPro.hashCode();
        return hash;
    }
}

```

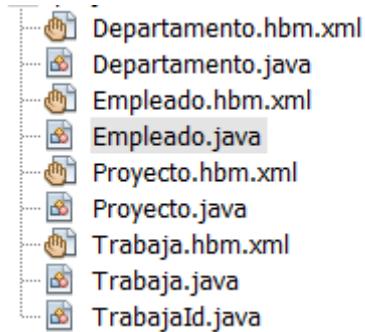
A la hora de almacenar información se relaciona con una base de datos. Por tanto, la mayoría de los objetos se encargan de mapearlos.

Las bases de datos son relacional. La mayoría de los objetos se encargan de mapearlos.

El mapeo objeto-relacional es necesario para trabajar con una base de datos.

El ORM se encarga de mapear los objetos.

En el siguiente enlace encontrarás un documento muy interesante sobre la problemática asociada al almacenamiento de objetos y al concepto de mapeo objeto-relacional.



Podemos observar que el asistente de **Hibernate** nos ha creado las clases **Departamento**, **Empleado**, **Proyecto**, **Trabaja**, **Trabajaid**.

```

empleado
cod_emp CHAR(3)
nom_emp VARCHAR(30)
fecha_ingreso DATE
salario FLOAT(6,2)
cod_jefe CHAR(3)
cod_dept CHAR(2)
Indexes

public class Empleado implements java.io.Serializable {
    private String codEmp;
    private Departamento departamento;
    private Empleado empleado;
    private String nomEmp;
    private Date fechaIngreso;
    private Float salario;
    private Set empleados = new HashSet(0);
    private Set trabajas = new HashSet(0);
}

```

Observamos que la clase generada **Empleado** tiene atributos que tienen su correspondencia con la tabla **Empleado** de la bbdd, como por ejemplo **codEmp**, **nomEmp**, **fechaIngreso**, **salario**, **cod_jefe**, **cod_dept**, podemos ver que en algunos casos solo ha cambiado su nombre brevemente quitando el guion bajo.

En la clase observamos también varias colecciones de datos **HashSet**, como **empleados** y **trabajas**.

```

public Set getEmpleados() {
    return this.empleados;
}

public void setEmpleados(Set empleados) {
    this.empleados = empleados;
}

public Set getTrabajas() {
    return this.trabajas;
}

public void setTrabajas(Set trabajas) {
    this.trabajas = trabajas;
}

```

Estas instancias de colección **HashSet** se distinguen por contener la clave foránea de la entidad que posee la colección. Con **HashSet** se mapea la columna clave de la colección.

2.1.2 ¿Cambia algo en el fichero de configuración de Hibernate?

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/
3 <!--
4 <!-->
5 <!-->
6 <!-->
7 <!-->
8 <!-->
9 <!-->
10 <!-->
11 <!-->
12 <!-->
13 <!-->
14 <!-->
15 <!-->
16 <!-->
17 <!-->
```

En el fichero **hibernate.cfg** se han añadido los parámetros “**mapping**” que mapean las distintas tablas de las entidades de la bbdd en clases de Java.

El archivo de configuración **Hibernate.cfg.xml** además de la información sobre la conexión de base de datos, contiene las asignaciones de los recursos y otras propiedades de conexión.

2.1.3 ¿Qué ocurre con los nombres de los atributos de las clases (por ejemplo: nom_dept)?

```
/*
 * Departamento generated by hbm2java
 */
public class Departamento implements java.io.Serializable {

    private String codDept;
    private String nomDept;
    private String ciudad;
    private Set empleados = new HashSet(0);
    private Set proyectos = new HashSet(0);

    public Departamento() {
    }
}
```

Nom_depto se ha modificado por el atributo nomDept sin guion bajo

2.1.4 ¿Se llaman igual que los atributos de las tablas?

Tenemos algunos atributos que se llaman igual en cambio hay otros que tienen otro nombre.

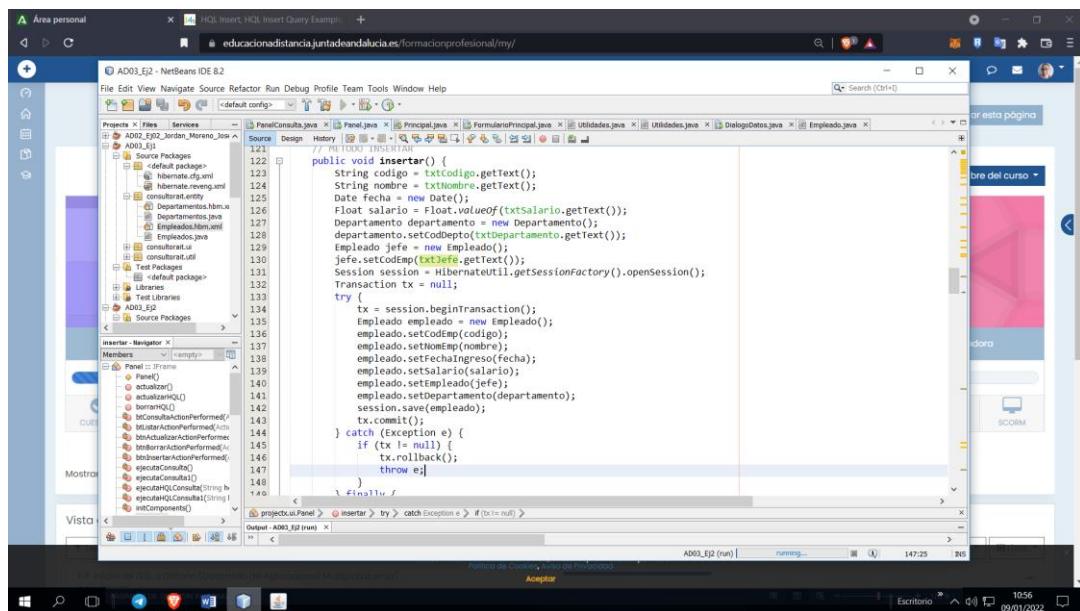
Las **clases persistentes** tienen la capacidad de definir objetos que pueden almacenarse y recuperarse, son un almacén persistente de datos. La especificación **JDO** incorpora la figura del procesador de clases en código ejecutable Java, **JDO Enhacer**, que es un programa que **modifica los archivos compilados de las clases**, añadiendo el código ejecutable necesario para realizar la grabación y recuperación transparente de los atributos de las instancias persistentes.

JDO permite convertir las clases en persistentes, de forma que los objetos pertenecientes a clases concretas definidas pueden mantener su estado, con la única limitación de que el estado esté compuesto por los **atributos persistentes** que sean independientes del contexto de ejecución: tipos primitivos, tipos de referencia e interfaz y algunas clases del sistema que permiten modelar el estado como por ejemplo la clase Array, Date, etc.

Para poder indicar las clases y atributos que son persistentes, se utiliza un fichero de configuración XML, que se denomina **descriptor de persistencia**. Para que las instancias de las clases persistentes pueden mantenerse en los sistemas gestores de bases de datos, **es necesario establecer la correspondencia entre los objetos y su estado persistente.**

2.2 Crea y añade a tu proyecto los siguientes métodos:

2.2.1 Inserta un nuevo objeto Empleados en la BD (código empleado: 'A22', nombre: 'Pepe García', fecha de ingreso: 'la fecha actual', salario: 2000, jefe: 'A11' y departamento '01'). Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha insertado correctamente.

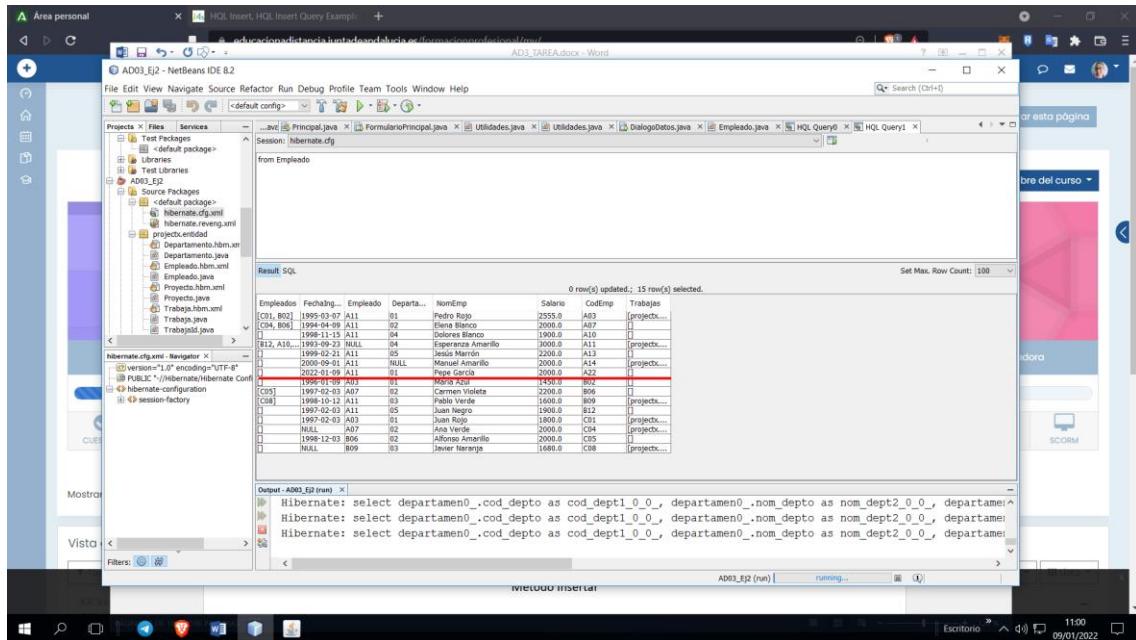


```

    // ÁREA PERSONAL
    // HQL Insert: HQL Insert Query Example
    // educacionadistancia.juntadeandalucia.es/formacionprofesional/my/
    // AD03_Ej2 - NetBeans IDE 8.2
    // File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
    // Search (Ctrl+F)
    // Edición: /src/panel/consultar/insertar
    public void insertar() {
        String codigo = txtCodigo.getText();
        String nombre = txtNombre.getText();
        Date fecha = new Date();
        Float salario = Float.valueOf(txtSalario.getText());
        Departamento departamento = new Departamento();
        departamento.setCoddepto(txtDepartamento.getText());
        Empleado jefe = new Empleado();
        jefe.setNombre(txtJefe.getText());
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = null;
        try {
            tx = session.beginTransaction();
            Empleado empleado = new Empleado();
            empleado.setCodemp(codigo);
            empleado.setNombre(nombre);
            empleado.setFechaingreso(fecha);
            empleado.setSalario(salario);
            empleado.setJefe(jefe);
            empleado.setDepartamento(departamento);
            session.save(empleado);
            tx.commit();
        } catch (Exception e) {
            if (tx != null) {
                tx.rollback();
            }
            throw e;
        }
    }
}

```

Método Insertar



Consulta HQL

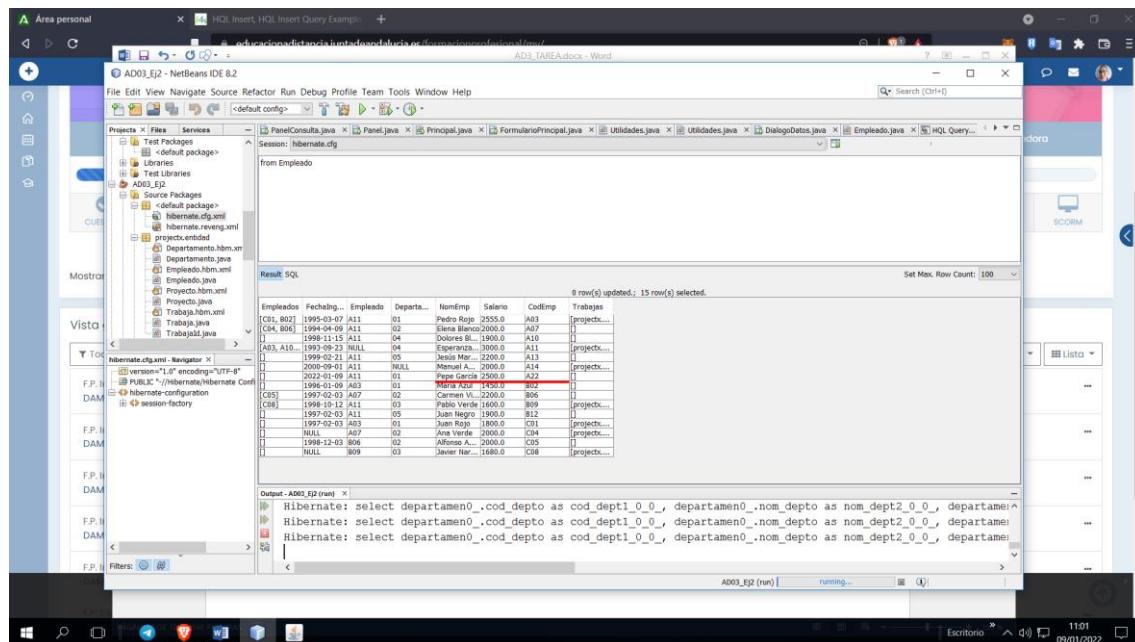
2.2.2 Modifica el salario del objeto Empleados de la BD creado en el apartado anterior (empleado A22) con un nuevo valor de 2500. Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha modificado correctamente.

```
//METODO ACTUALIZAR HQL
public void actualizarHQL() {
    String codigo = txtCodigo.getText();
    Float salario = Float.valueOf(txtSalario.getText());
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();
        //session.createQuery("update Empleado set salario=" + salario + " where codEmp=" + "'" + codigo + "'").executeUpdate();
        session.createQuery("update Empleado set salario=" + salario + " where codEmp=" + "'" + codigo + "'").executeUpdate();
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
            throw e;
        }
    } finally {
        session.close();
    }
}
```

Método actualizar con HQL

```
//METODO ACTUALIZAR con OBJETOS
public void actualizar() {
    String codigo = txtCodigo.getText();
    Float salario = Float.valueOf(txtSalario.getText());
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();
        Empleado empleado = (Empleado) session.get(Empleado.class, codigo);
        empleado.setSalario(salario);
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
            throw e;
        }
    } finally {
        session.close();
    }
}
```

Método actualizar con objetos



Consulta HQL

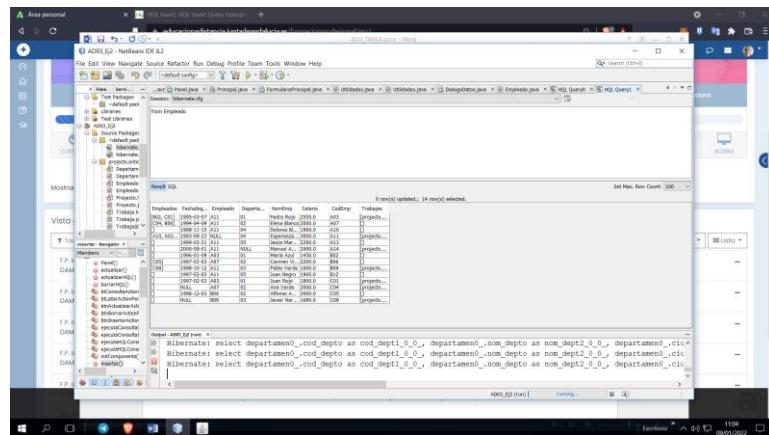
2.2.3 Elimina un objeto Empleados de la BD (por ejemplo el empleado A22). Ejecuta una consulta desde el intérprete de HQL incluido en NetBeans para comprobar que se ha eliminado correctamente.

```

199
200
201     //METODO BORRAR HQL
202     public void borrarHQL() {
203         String codigo = txtCodigo.getText();
204         Session session = HibernateUtil.getSessionFactory().openSession();
205         Transaction tx = null;
206         try {
207             tx = session.beginTransaction();
208             session.createQuery("delete from Empleado where codEmp=" + "'" + codigo + "'").executeUpdate();
209             tx.commit();
210         } catch (Exception e) {
211             if (tx != null) {
212                 tx.rollback();
213                 throw e;
214             }
215         } finally {
216             session.close();
217         }
218     }

```

Método borrarHQL



Consulta HQL, comprobamos que se ha eliminado el objeto requerido.

2.2.4 Realiza (codifica) la siguiente consulta: "Empleados con salario superior a 1000 € (mostrar su nombre, número de departamento y salario)".

Consulta en HQL : “ from Empleado where salario > 1000”

```
//Consulta: Empleados con salario superior a 1000 € (mostrar su nombre, número de departamento y salario).
private static String CONSULTA1 = "from Empleado where salario > 1000";
```

Nombre	Salario	Departamento
Pedro Rojo	2555.0	01
Elena Blanco	2000.0	02
Dolores Blanco	1900.0	04
Esperanza Amarillo	3000.0	04
Jesús Marrón	2200.0	05
Manuel Amarillo	2000.0	
Maria Azul	1450.0	01
Carmen Violeta	2200.0	02
Pablo Verde	1600.0	03
Juan Negro	1900.0	05
Juan Rojo	1800.0	01
Ana Verde	2000.0	02

Consulta: "Empleados con salario superior a 1000 €"

Salida en la aplicación Java

```

from Empleado where salario > 1000

```

Empleado	FechaNaci...	Empleado	Departa...	Nombre	Salario	CodEmp	Trabajos
[002, C01]	1995-03-07	A11	01	Pedro Rojo	2555.0	A03	[projecte...]
[C04, B06]	1994-04-09	A11	02	Elena Blanco	2000.0	A07	[]
[J1, A01]	1994-04-09	A11	02	Dolores Blan...	1900.0	A10	[]
[A14, B05]	1993-09-23	NULL	04	Esperanza Amari...	3000.0	A11	[projecte...]
[]	1999-02-21	A11	05	Jesús Marrón	2200.0	A13	[]
[]	1999-02-21	A11	05	Manuel Amari...	2000.0	A14	[]
[]	1996-01-09	A03	01	Maria Azul	1450.0	B02	[]
[C05]	1997-02-03	A07	02	Carmen Vl...	2200.0	B06	[]
[C08]	1997-02-03	A07	02	Juan Negro	1900.0	B12	[]
[]	1997-02-03	A11	05	Juan Rojo	1800.0	C01	[projecte...]
[]	1997-02-03	A03	01	Ana Verde	2000.0	C04	[projecte...]
[]	1998-12-03	B06	02	Alejandra A...	2000.0	C05	[]
[]	NULL	B09	03	Javier Nar...	1600.0	C08	[projecte...]

Output - AD03_Ej2 [run] x

- run t
- BUILD SUCCESSFUL (total time: 2 seconds)

Consulta HQL

