



CICLO: [DAM]
MÓDULO DE [ACCESO A DATOS]

[Tarea N° 4]

Alumno:
[Jairo Martínez Garrido]
[76652856C]

Los documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos incluidos en este contenido pueden contener imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en el contenido. Fomento Ocupacional FOC SL puede realizar en cualquier momento, sin previo aviso, mejoras y/o cambios en el contenido.

Es responsabilidad del usuario el cumplimiento de todas las leyes de derechos de autor aplicables. Ningún elemento de este contenido (documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos asociados), ni parte de este contenido puede ser reproducida, almacenada o introducida en un sistema de recuperación, ni transmitida de ninguna forma ni por ningún medio (ya sea electrónico, mecánico, por fotocopia, grabación o de otra manera), ni con ningún propósito, sin la previa autorización por escrito de Fomento Ocupacional FOC SL.

Este contenido está protegido por la ley de propiedad intelectual e industrial. Pertenecen a Fomento Ocupacional FOC SL los derechos de autor y los demás derechos de propiedad intelectual e industrial sobre este contenido.

Sin perjuicio de los casos en que la ley aplicable prohíbe la exclusión de la responsabilidad por daños, Fomento Ocupacional FOC SL no se responsabiliza en ningún caso de daños indirectos, sean cuales fueren su naturaleza u origen, que se deriven o de otro modo estén relacionados con el uso de este contenido.

©2018 Fomento Ocupacional FOC SL todos los derechos reservados.

Contenido

1. Documentos que se adjuntan a este informe.....2
2. Resto de epígrafes que componen el desarrollo de este informe.....**¡Error! Marcador no definido.**
3. Componentes del grupo (solo en la tarea colaborativa).14

(Una vez realizado el informe, no olvidar actualizar esta tabla del índice **(F9 + Actualizar toda la tabla)**, con el fin de que se actualicen todos los epígrafes y números de página)

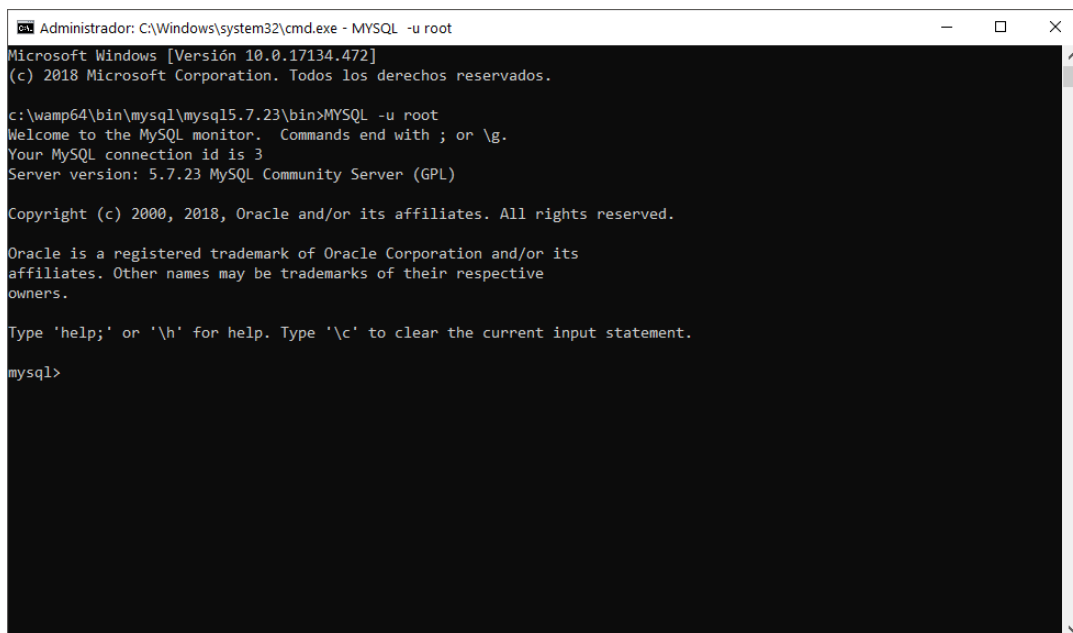
1. Documentos que se adjuntan a este informe.

A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2.

2. Configuración del proyecto HIBERNATE

Creación de la base de datos



```
Administrador: C:\Windows\system32\cmd.exe - MYSQL -u root
Microsoft Windows [Versión 10.0.17134.472]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

c:\wamp64\bin\mysql\mysql5.7.23\bin>MYSQL -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

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>
```

Conectamos a MYSQL mediante el comando **MYSQL -u root**

```
mysql> CREATE DATABASE gestionLaboral1819;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Creamos la base de datos **gestionlaboral1819**

```
mysql> USE gestionlaboral1819;
Database changed
mysql> ■
```

Seleccionamos la base de datos con la que vamos a trabajar.

```
mysql> CREATE TABLE empleados(
  -> CodEmpleado INT PRIMARY KEY,
  -> Nombre VARCHAR(50),
  -> Apellidos VARCHAR(50),
  -> Puesto VARCHAR(100),
  -> Salario FLOAT(6,2)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql> █
```

Introducimos los datos de los 5 empleados

```
mysql> INSERT INTO empleados (CodEmpleado, Nombre, Apellidos, Puesto, Salario) VALUES (1,"Juan","Martín","Administrativo",1200);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO empleados (CodEmpleado, Nombre, Apellidos, Puesto, Salario) VALUES (2,"Ana","Amezcue","Programador Senior",1500);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO empleados (CodEmpleado, Nombre, Apellidos, Puesto, Salario) VALUES (3, "Raúl","Perez", "Programador Junior", 1400);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO empleados (CodEmpleado, Nombre, Apellidos, Puesto, Salario) VALUES (4, "Jairo", "Martínez", "Programador Junio", 1400);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO empleados (CodEmpleado, Nombre, Apellidos, Puesto, Salario) VALUES (5, "Celia", "Carrillo", "Jefe de equipo", 1800);
Query OK, 1 row affected (0.00 sec)
```

Mostramos toda la tabla tras introducir los empleados.

```
mysql> SELECT * FROM empleados;
+-----+-----+-----+-----+-----+
| CodEmpleado | Nombre | Apellidos | Puesto | Salario |
+-----+-----+-----+-----+-----+
| 1 | Juan | Martín | Administrativo | 1200.00 |
| 2 | Ana | Amezcue | Programador Senior | 1500.00 |
| 3 | Raúl | Perez | Programador Junior | 1400.00 |
| 4 | Jairo | Martínez | Programador Junio | 1400.00 |
| 5 | Celia | Carrillo | Jefe de equipo | 1800.00 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

Damos **permisos** al usuario **foc** cuya contraseña es **fomento2018**

```
mysql> GRANT ALL ON tienda.*TO foc@localhost IDENTIFIED BY "fomento2018";  
Query OK, 0 rows affected, 1 warning (0.01 sec)  
  
mysql>
```

Creación de aplicación consulta_BD_Hibernate

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

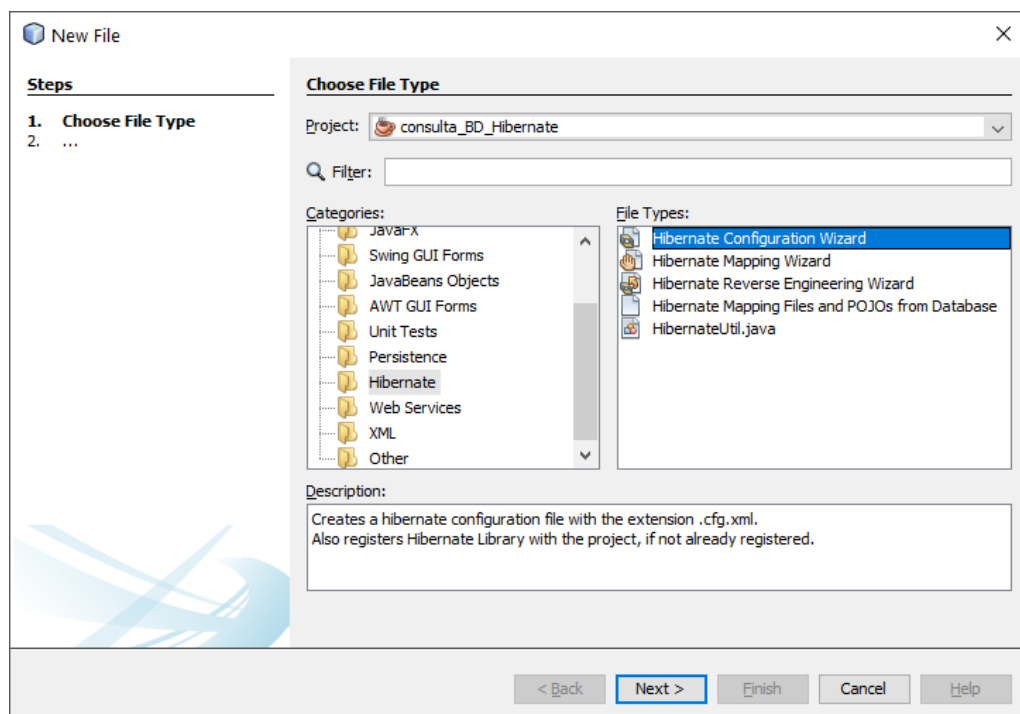
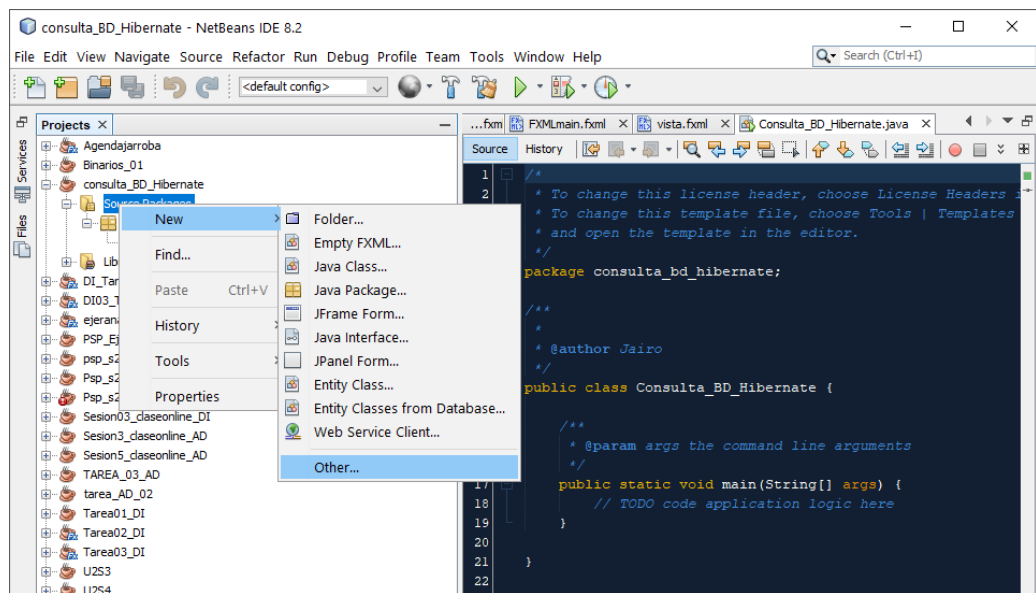
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back Next > **Finish** Cancel Help

Configuración de Hibernate.



New Hibernate Configuration Wizard

Steps

1. Choose File Type
- 2. Name and Location**
3. Select Data Source

Name and Location

File Name:

Project:

Folder:

Created File:

< Back Next > Finish Cancel Help

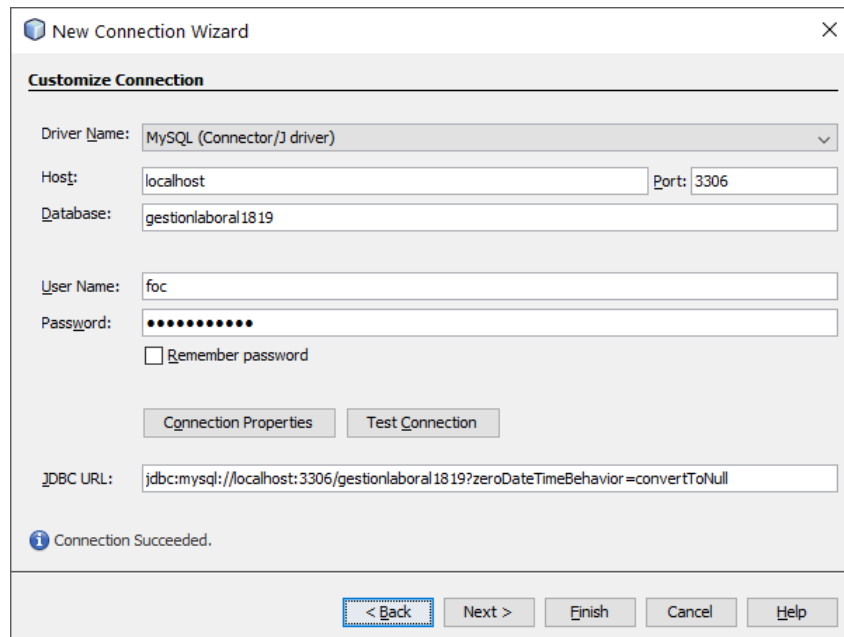
New Connection Wizard

Locate Driver

Driver:

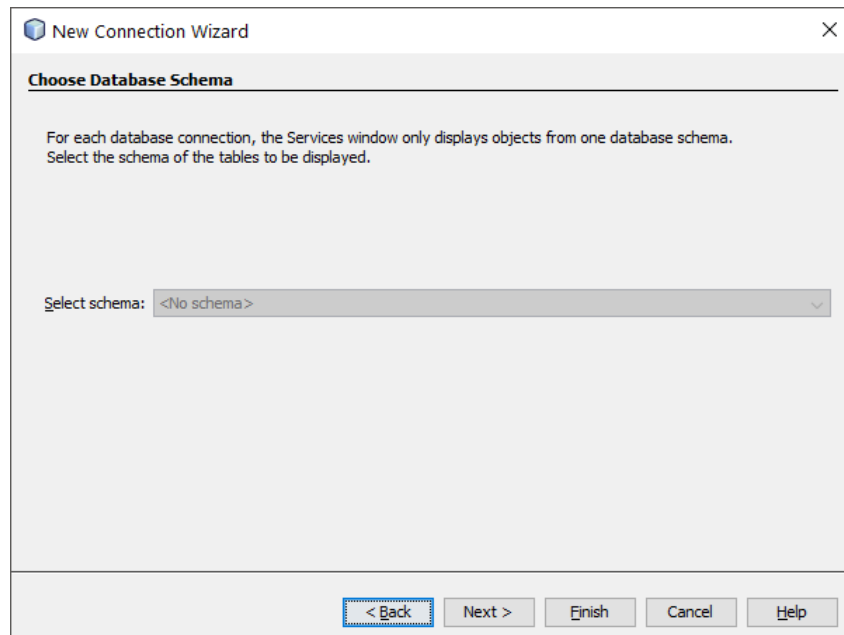
Driver File(s):

< Back Next > Finish Cancel Help



The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Customize Connection' step. The dialog has a title bar with a close button (X). Below the title bar is a section header 'Customize Connection'. The main area contains several fields: 'Driver Name' is a dropdown menu set to 'MySQL (Connector/J driver)'; 'Host' is a text box with 'localhost'; 'Port' is a text box with '3306'; 'Database' is a text box with 'gestionlaboral1819'; 'User Name' is a text box with 'foc'; 'Password' is a text box with masked characters (dots); and 'Remember password' is an unchecked checkbox. Below these fields are two buttons: 'Connection Properties' and 'Test Connection'. At the bottom, there is a 'JDBC URL' text box containing 'jdbc:mysql://localhost:3306/gestionlaboral1819?zeroDateTimeBehavior=convertToNull'. A status bar at the bottom left shows an information icon and the text 'Connection Succeeded.'. At the bottom right, there are five buttons: '< Back' (highlighted with a blue dashed border), 'Next >', 'Finish', 'Cancel', and 'Help'.

Configuramos la nueva conexión introduciendo el nombre de la base de datos, el usuario y la contraseña (**DB:** gestionlaboral1819, **User:** foc, **Password:** fomento2018) y probamos la conexión pulsando el botón **Test Connection**.



The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Choose Database Schema' step. The dialog has a title bar with a close button (X). Below the title bar is a section header 'Choose Database Schema'. The main area contains a text box with the following text: 'For each database connection, the Services window only displays objects from one database schema. Select the schema of the tables to be displayed.' Below this text is a 'Select schema:' label followed by a dropdown menu set to '<No schema>'. At the bottom right, there are five buttons: '< Back' (highlighted with a blue dashed border), 'Next >', 'Finish', 'Cancel', and 'Help'.

New Connection Wizard

Choose name for connection

Override the default name for the connection. The name should be descriptive about the connection you are creating.

Input connection name:

`jdbc:mysql://localhost:3306/gestionlaboral1819?zeroDateTimeBehavior=convertToNull [foc on Default schema]`

< Back Next > **Finish** Cancel Help

New Hibernate Configuration Wizard

Steps

1. Choose File Type
2. Name and Location
3. **Select Data Source**

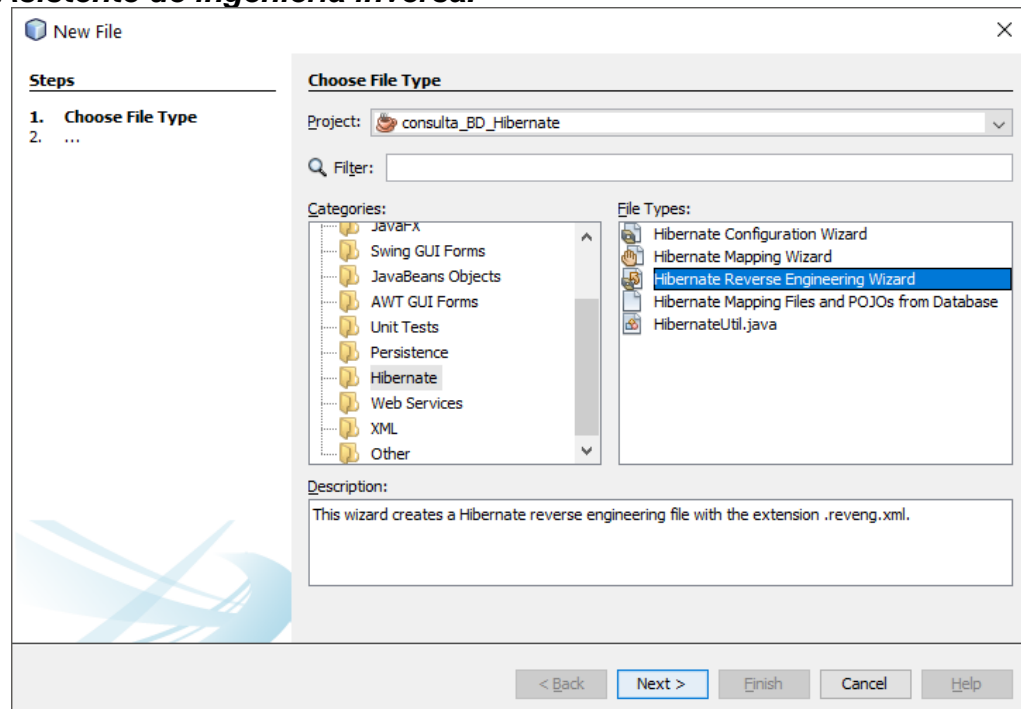
Select Data Source

Database Connection: `jdbc:mysql://localhost:3306/gestionlaboral1819?zeroDateTimeBehavior=c...`

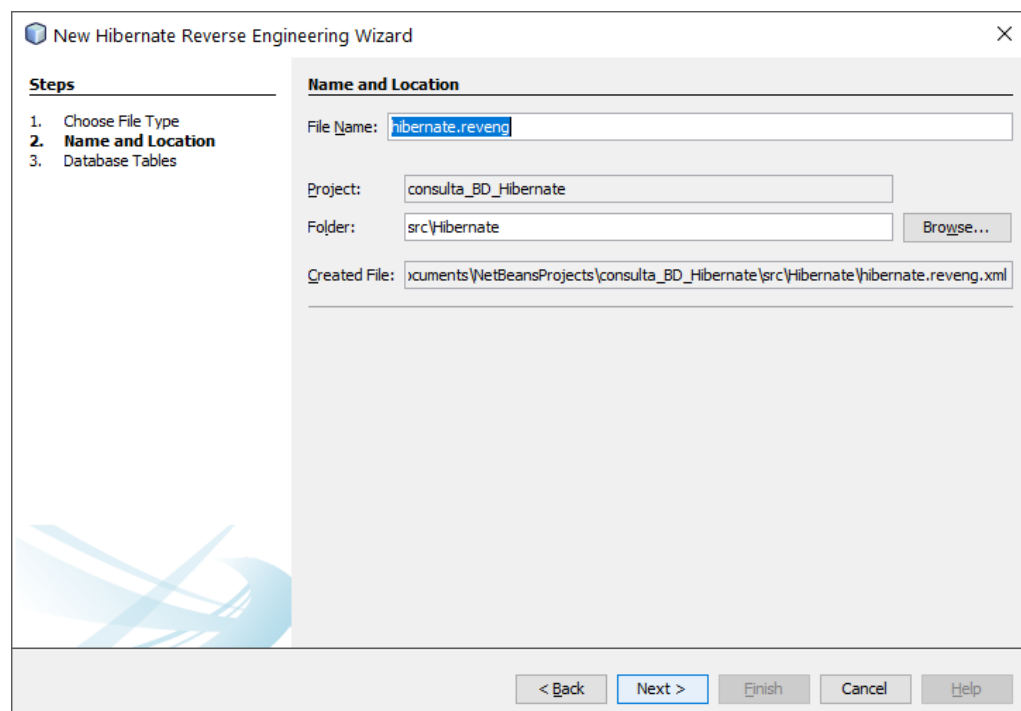
Database Dialect: `org.hibernate.dialect.MySQLDialect`

< Back Next > **Finish** Cancel Help

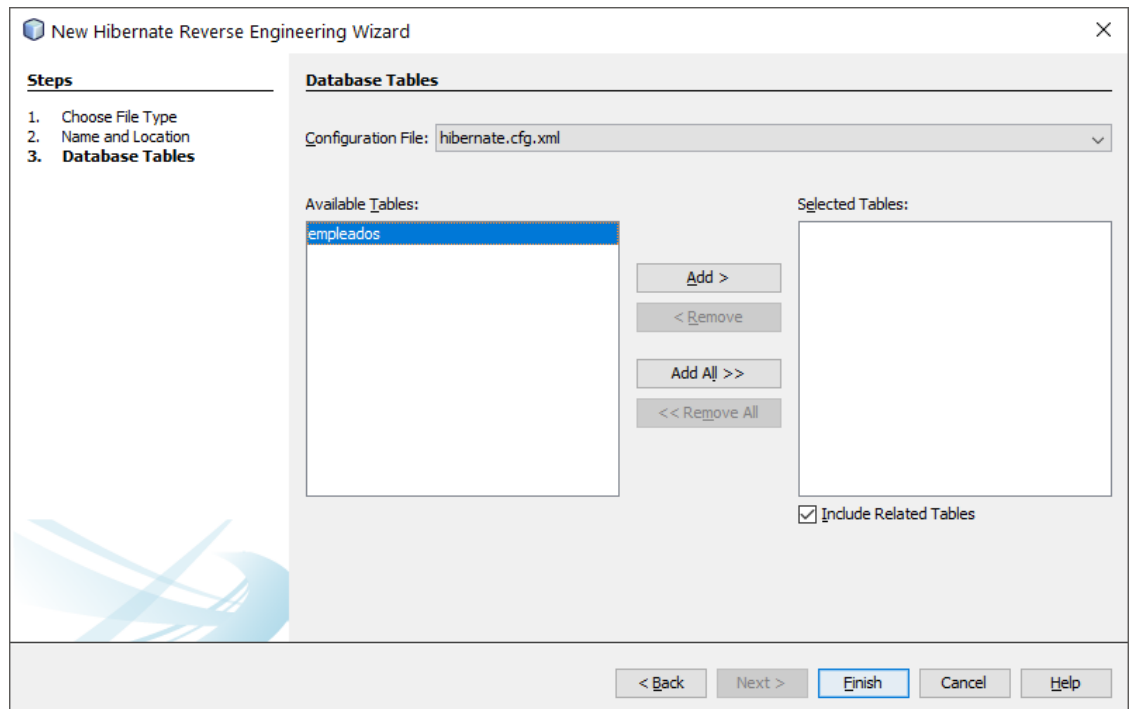
```
<hibernate-configuration>
<session-factory>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/gestionlaboral1819?zeroDateTimeBehavior=convertToNull</property>
  <property name="hibernate.connection.username">foc</property>
  <property name="hibernate.connection.password">Fomento2018</property>
</session-factory>
</hibernate-configuration>
```

Asistente de ingeniería inversa.

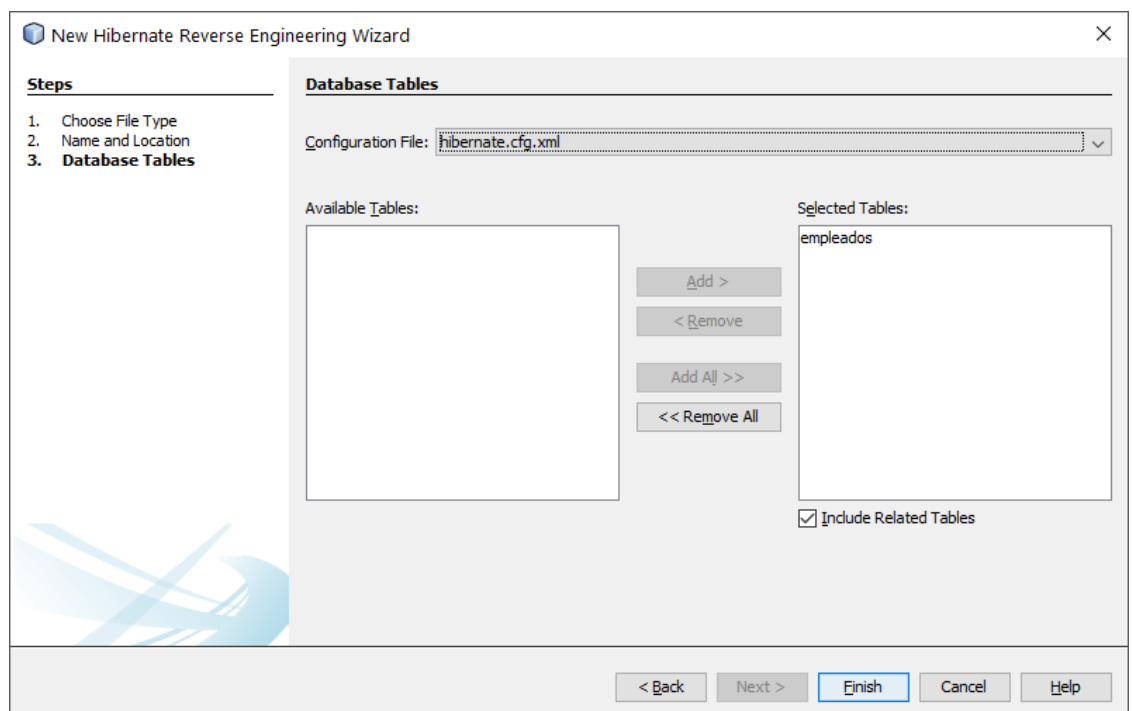
Seleccionamos Hibernate Reverse Engineering Wizard (asistente).



Pulsamos Next>



Seleccionamos **empleados** y pulsamos **Add>**



Una vez que la tabla se encuentra seleccionada, pulsamos **Finish**

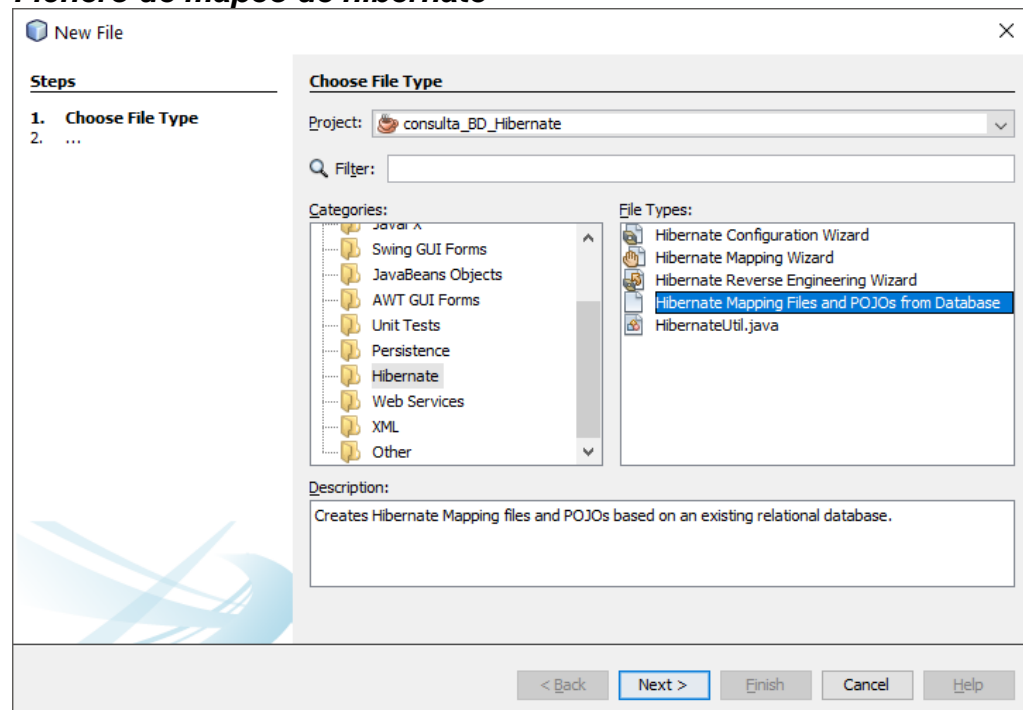
```

1
2 <hibernate-reverse-engineering>
3   <schema-selection match-catalog="gestionlaboral1819"/>
4   <table-filter match-name="empleados"/>
5 </hibernate-reverse-engineering>
6

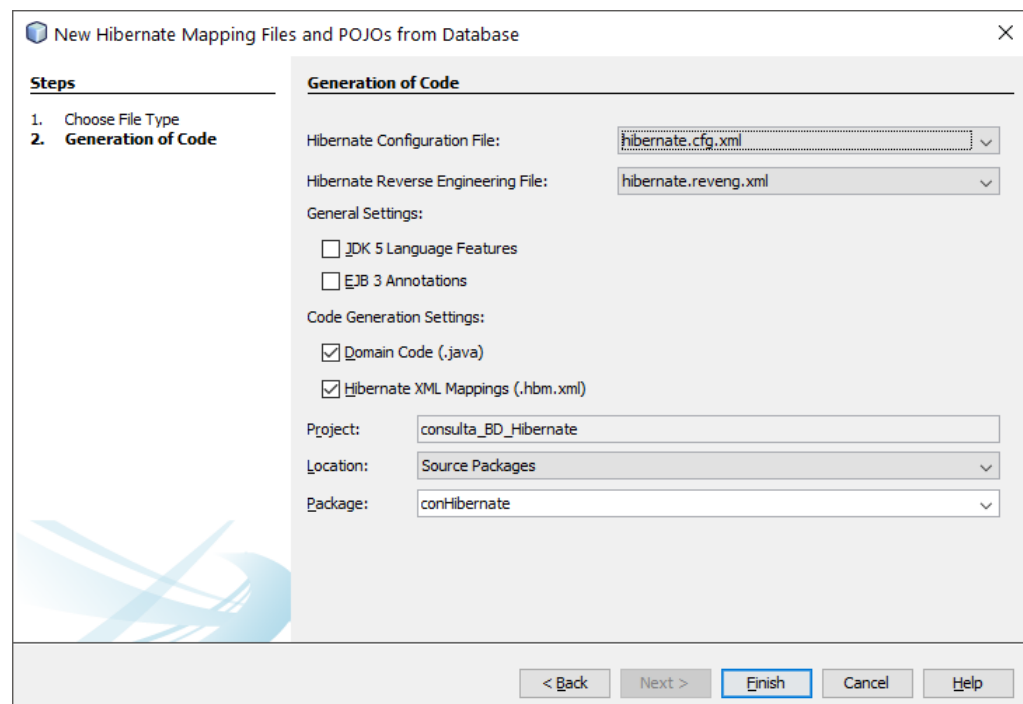
```

Este es el contenido del archivo hibernate reverse engineering.

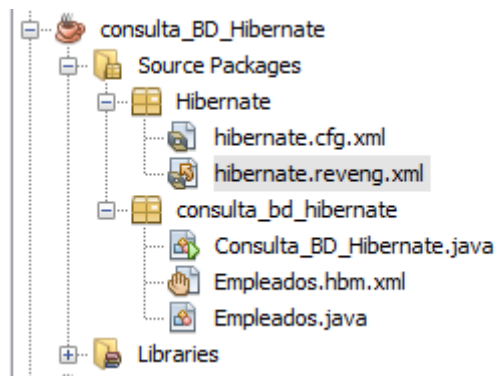
Fichero de mapeo de hibernate



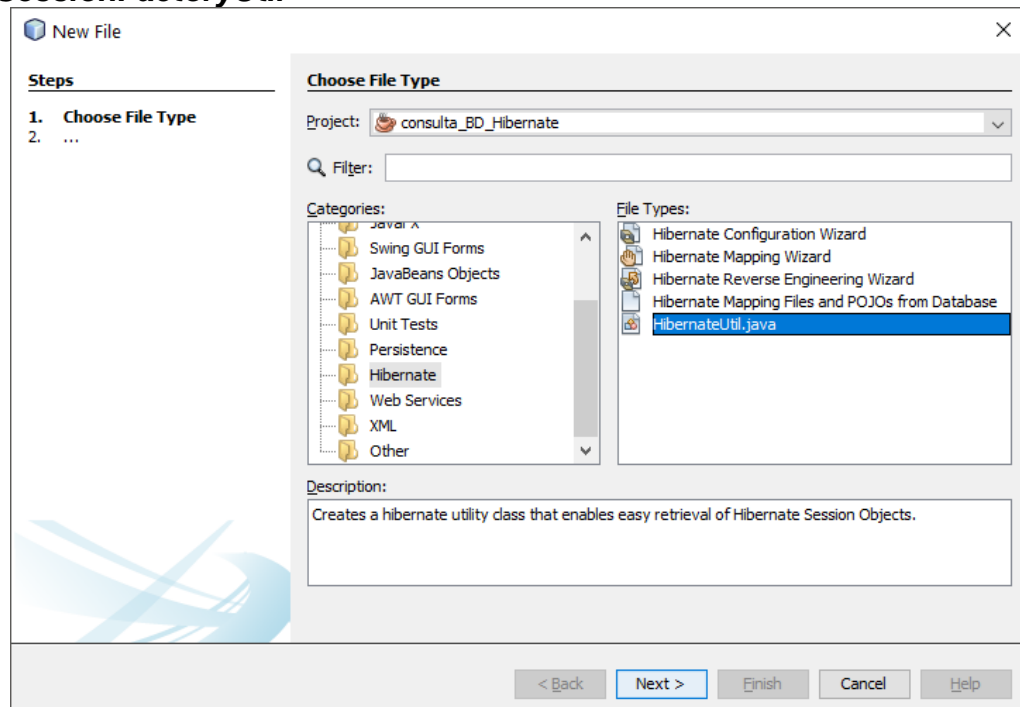
Seleccionamos Hibernate Mapping Files and POJOs from Database y pulsamos Next>.



Comprobamos que el archivo configuration y reverse engineering son correctos y pulsamos Finish



SessionFactoryUtil



Seleccionamos HibernateUtil y pulsamos Next>

New HibernateUtil.java

Steps

1. Choose File Type
2. Name and Location

Name and Location

Class Name: SessionFactorySingleton

Project: consulta_BD_Hibernate

Location: Source Packages

Package: consulta_bd_hibernate

Created File: objects\consulta_BD_Hibernate\src\consulta_bd_hibernate\SessionFactorySingleton.java

< Back Next > Finish Cancel Help

Cambiamos el nombre por el deseado, en nuestro caso **SessionFactoryUtil** y pulsamos Finish.

```
public class SessionFactorySingleton {

    private static final SessionFactory sessionFactory;

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

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

}
```

El contenido del archivo debe ser como el que podemos observar en la imagen.

3. Clase consultarEmpleadosHIBERNATE.

En esta clase crearemos una sesión con la que accederemos a la base de datos gestionlaboral1819 y consultaremos la información de los empleados de la tabla Empleados.

```
public class consultarEmpleadosHIBERNATE {

    public static void main(String[] args) {
        //Creamos una sesion y trabajamos con ella
        SessionFactory sessionGeneral = SessionFactoryUtil.getSessionFactory();
        Session sessionActual = sessionGeneral.openSession();
```

Creamos un objeto SessionFactory que contendrá un objeto configuration con la información relacionada con la Base de Datos. El objeto SessionFactory nos permitirá crear una sesión de conexión con la base de datos que estemos utilizando. Para llevarlo a cabo obtenemos el SessionFactory que se ha creado para esta sesión y lo almacenamos en sessionGeneral.

A continuación creamos un objeto **Session** cuyo nombre es **sesionActual**, al cual le indicamos que inicie la transacción con la base de datos mediante **.openSession()**.

```
//Establecemos la sentencia
Query q = sessionActual.createQuery("from Empleados");

//Asignamos los resultados a un Array
List<Empleados> listaEmpleados = q.list();
```

El siguiente es el uso de la interfaz **Query** para crear una sentencia de consulta en la sesión actual. Tras ello creamos un Array tipo **List<Empleados>** que almacenará la información recogida mediante la consulta anteriormente realizada.

```
System.out.println("");
System.out.println("");
System.out.println("INFORMACIÓN DE LOS EMPLEADOS");
System.out.println("=====");

//Recorremos el ArrayList de forma clásica
for(Empleados empleadoActual : listaEmpleados){
    System.out.println("codEmpleado: "+empleadoActual.getCodEmpleado()
        + "\tNombre: "+empleadoActual.getNombre()+" "
        + "\tApellido: "+empleadoActual.getApellidos()+" "
        + "\tPuesto: "+empleadoActual.getPuesto()+" "
        + "\t\tSalario: "+empleadoActual.getSalario());
}
System.out.println("=====");
sessionActual.close();
}
```

El último paso de esta clase es la impresión de los datos almacenados en **listaEmpleados**, para ello creamos un bucle **for-each** en el que le indicamos que para cada objeto

empleadoActual de tipo **Empleado** del array **listaEmpleados** realice la acción de imprimir los atributos solicitados mediante los métodos **getCodEmpleado()**, **getNombre()**, **getApellidos()**, **getPuesto()** y **getSalario()**

Impresión por consola

INFORMACIÓN DE LOS EMPLEADOS

```
=====
codEmpleado: 1  Nombre: Juan  Apellido: Martín  Puesto: Administrativo  Salario: 1200.0
codEmpleado: 2  Nombre: Ana  Apellido: Amezcu  Puesto: Programador Senior  Salario: 1500.0
codEmpleado: 3  Nombre: Raúl  Apellido: Perez  Puesto: Programador Junior  Salario: 1400.0
codEmpleado: 4  Nombre: Jairo  Apellido: Martínez  Puesto: Programador Junior  Salario: 1400.0
codEmpleado: 5  Nombre: Celia  Apellido: Carrillo  Puesto: Jefe de equipo  Salario: 1800.0
=====
```

4. Clase consultarPuestosHIBERNATE.

Esta clase ejecuta un menú en el cual debemos elegir las acciones que queremos realizar, ofreciéndonos la posibilidad de consultar la información de los empleados pertenecientes a cada puesto laboral.

```
public class consultarPuestosHIBERNATE {
    public static void main(String[] args) {

        //Creamos una sesión y trabajamos con ella
        SessionFactory sessionGeneral = SessionFactoryUtil.getSessionFactory();
        Session sessionActual = sessionGeneral.openSession();
    }
}
```

Al igual que en la clase anterior, creamos un objeto **SessionFactory** que almacenará la **configuración de la base de datos** y con el cual podremos obtener nuestra SessionFactory para posteriormente crear un objeto **Session** a través del cual iniciar la transacción con la base de datos.

```
Scanner scanner = new Scanner(System.in);
boolean salir = false;
```

El siguiente paso es crear dos elementos que necesitaremos para la ejecución de nuestro menú, un **Scanner** para recoger la información introducida por teclado y una **variable de tipo boolean** que utilizaremos como **interruptor** de tipo lógico de nuestro menú, lo que nos permitirá salir de él cuando hayamos terminado de consultar la información.

```
Query q;
List<Empleados> listaEmpleados;
```

También preparamos nuestro objeto de tipo **Query**, con el cual realizaremos las sentencias de consulta posteriormente, y creamos nuestro array **List<Empleados>** que denominamos **listaEmpleados**.

A continuación creamos un bucle **do-while** que almacenará nuestro menú.

```
do{
    System.out.println("¿Desea consultar los diferentes puestos de trabajo?");
    System.out.println("1 - Si");
    System.out.println("2 - Salir\n");
    System.out.println("Introduzca el número de la respuesta: ");
    final int sc = scanner.nextInt();
```

La primera parte de nuestro menú consiste en la impresión de la información necesaria para que el usuario sea capaz de interactuar, indicamos que si introduce 1 acepta continuar y si introduce 2 saldrá del menú. Además, creamos nuestra variable final de tipo int **sc** en la que almacenaremos la información recogida mediante el **Scanner** anteriormente creado.

A continuación, declaramos un bloque **try-catch**, lo hacemos así porque dentro del bloque try ubicaremos todo el cuerpo de código que es susceptible de generar errores.

```
try{
    if(sc == 1){
        q = sessionActual.createQuery("from Empleados");
        listaEmpleados = q.list();

        System.out.println("=====");
        for(Empleados emp: listaEmpleados){
            System.out.println("- "+emp.getPuesto());
        }
    }
```

Dentro del bloque try creamos un condicional **if-elseif**. En la condicional if establecemos que si el valor recogido por el Scanner y almacenado en la variable **sc** es **1**, se realice la sentencia de consulta mediante **.createQuery()** y se almacene en nuestro objeto **Query q**, para a continuación guardar esta información dentro del array **listaEmpleados** mediante el método **.list()**.

El siguiente paso es proceder a mostrar todos los puestos que existen dentro de la tabla Empleados mediante un **for-each** que recorra la **listaEmpleados** y acceda a los puestos

mediante el método **getPuestos()** al que podemos acceder desde el objeto Empleados **emp**

```
final int scSwitch = scanner.nextInt();
System.out.println("");
```

Creamos una nueva variable **final int scSwitch** que recoja la información del scanner, la cual utilizaremos como condicional para el **switch**.

```
switch(scSwitch){
    case 1:
        System.out.println("Has seleccionado: Administrativo");
        System.out.println("-----\n");
        q = sesionActual.createQuery("from Empleados emp where emp.puesto='Administrativo'");
        listaEmpleados = q.list();
        consultaPuestos(listaEmpleados);
        break;
    case 2:
        System.out.println("Has seleccionado Programador Senior");
        System.out.println("-----\n");
        q = sesionActual.createQuery("from Empleados emp where emp.puesto='Programador Senior'");
        listaEmpleados = q.list();
        consultaPuestos(listaEmpleados);
        break;
    case 3:
        System.out.println("Has seleccionado Programador Junior");
        System.out.println("-----\n");
        q = sesionActual.createQuery("from Empleados emp where emp.puesto='Programador Junior'");
        listaEmpleados = q.list();
        consultaPuestos(listaEmpleados);
        break;
    case 4:
        System.out.println("Has seleccionado Jefe de equipo");
        System.out.println("-----\n");
        q = sesionActual.createQuery("from Empleados emp where emp.puesto='Jefe de equipo'");
        listaEmpleados = q.list();
        consultaPuestos(listaEmpleados);
        break;
    case 5:
        salir = true;
        break;
    default:
        System.out.println("***FIN DE LA APLICACIÓN***");
        salir=true;
        sesionActual.close();
        break;
}
```

Creamos un switch que reciba como parámetro condicional **scSwitch** e indicamos que para los valores comprendidos entre 1 y 5 tenga una respuesta. La estructura general de cada **case** realiza las siguientes acciones, primero aporta información al usuario sobre la opción que ha escogido mediante un **sout**, a continuación crea una sentencia de consulta a través de nuestro objeto Query **q**, el cual se transforma en lista mediante **.list()** y lo almacenamos en nuestro array **listaEmpleados**. El siguiente paso es llamar a la función **consultaPuestos()** que recibirá como parámetro **listaEmpleados**.

ConsultaPuestos()

```
public static void consultaPuestos(List<Empleados> list) {
    for (Empleados emp: list) {
        System.out.println("codEmpleado: "+emp.getCodEmpleado()+"\tNombre: "+emp.getNombre()+"\tApellido: "+emp.getApellidos()
            +"\tPuesto: "+emp.getPuesto()+"\tSalario: "+emp.getSalario());
    }
}
```

Se trata de un método sencillo que solo alberca un bucle **for-each** con el cual recorrer la **lista** que se le pasa como **parámetro** e imprime los detalles de los empleados que se han almacenado en la lista que hemos pasado como parámetros. Se trata de un método que nos **evita crear código duplicado**.

Las respuestas de los **case del 1 al 4 son similares**, cambiando entre ellas solo la consulta y la información obtenida que se almacena en **listaEmpleados**. Sin embargo, el **case "5"** cumple la función de pulsar nuestro interruptor lógico y cambiar el valor de la variable **salir** por **true** lo que hace finalizar nuestro bucle **do-while**. Además, añadimos un **default** que nos permite cerrar acabar la ejecución del menú, y cerrar la sesión de la base de datos.

```
}else if(sc==2){
    salir = true;
    sesionActual.close();
    System.out.println("***FIN DE LA APLICACIÓN***");
}
```

En el condicional **elseif** indicamos que si el valor almacenado en la **variable sc** es **2** se realicen las acciones necesarias para finalizar el proceso de la aplicación, cambiando nuestro **interruptor a true**, lo que finaliza nuestro bucle **do-while**, cerrar la transacción con la base de datos mediante el método **.close()** sobre nuestro objeto **Session sesionActual**, y además imprimimos un mensaje por consola indicando el fin de la aplicación para informar al usuario.

```
}catch(InputMismatchException e){
    System.out.println("Debes introducir correctamente las opciones");
    scanner.next();
}
}while(!salir);
}
```

En esta última imagen podemos ver que el bloque **catch** captura la excepción de tipo **InputMismatchException** e imprime un mensaje que indica que se debe introducir una opción correcta a través del scanner. También apreciamos que la condición de **while** indica que mientras la condición sea verdadera (recordamos que el valor de **salir** es **false**),

el bucle puede continuar, por lo que, al cambiar el valor de salir a **true**, la condición es falsa y se cierra el bucle.

Ejecución de la aplicación

Ejecución normal.

```
¿Desea consultar los diferentes puestos de trabajo?
1 - Si
2 - Salir

Introduzca el número de la respuesta:
1
=====
- Administrativo
- Programador Senior
- Programador Junior
- Programador Junior
- Jefe de equipo

Consultar empleados cuyo puesto de trabajo es
1- Administrativo
2- Programador Senior
3- Programador Junior
4- Jefe de equipo
5- Salir
Introduzca su respuesta:
1

Has seleccionado: Administrativo
-----

codEmpleado: 1  Nombre: Juan  Apellido: Martín  Puesto: Administrativo  Salario: 1200.0
¿Desea consultar los diferentes puestos de trabajo?
1 - Si
2 - Salir

Introduzca el número de la respuesta:
```

Introducción de valor erróneo

```

¿Desea consultar los diferentes puestos de trabajo?
1 - Sí
2 - Salir

Introduzca el número de la respuesta:
3
¿Desea consultar los diferentes puestos de trabajo?
1 - Sí
2 - Salir

Introduzca el número de la respuesta:
1
=====
- Administrativo
- Programador Senior
- Programador Junior
- Programador Junior
- Jefe de equipo

Consultar empleados cuyo puesto de trabajo es
1- Administrativo
2- Programador Senior
3- Programador Junior
4- Jefe de equipo
5- Salir
Introduzca su respuesta:
3

Has seleccionado Programador Junior
-----

codEmpleado: 3  Nombre: Raúl    Apellido: Perez Puesto: Programador Junior    Salario: 1400.0
codEmpleado: 4  Nombre: Jairo   Apellido: Martínez  Puesto: Programador Junior    Salario: 1400.0
¿Desea consultar los diferentes puestos de trabajo?
1 - Sí
2 - Salir

Introduzca el número de la respuesta:

```

Salir de la aplicación

```

- -
¿Desea consultar los diferentes puestos de trabajo?
1 - Sí
2 - Salir

Introduzca el número de la respuesta:
2
***FIN DE LA APLICACIÓN***
|

```