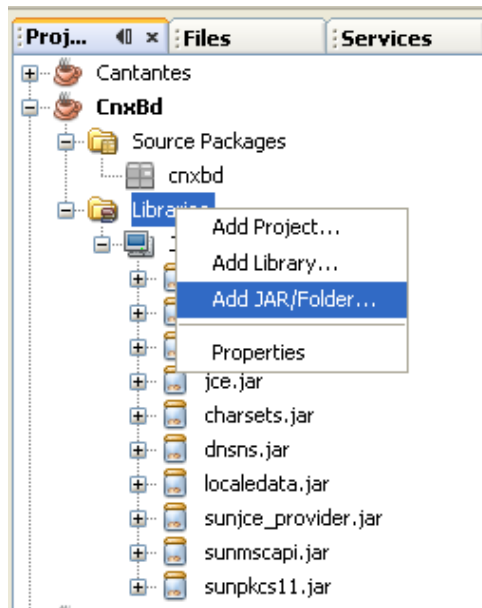


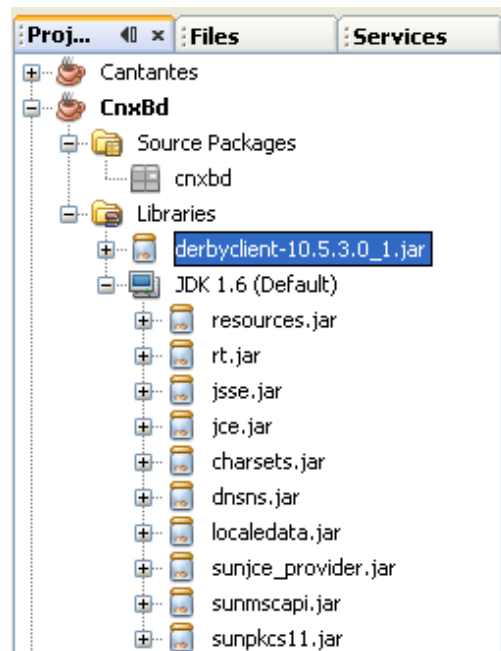


TALLER DE CONEXIÓN A BASES DE DATOS

1. Cree un nuevo proyecto Java Application
2. Descargue el respectivo driver para conectarse a una BD Derby. El driver JDBC para Derby: derbyclient.jar
3. Agregue el driver a la carpeta de librerías

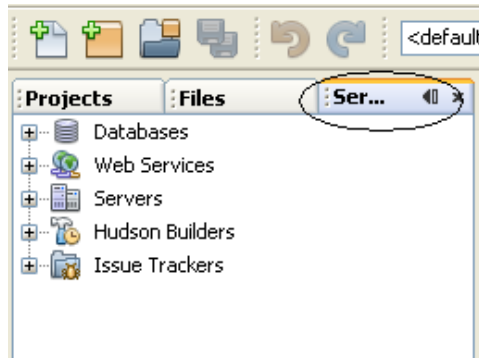


4. Después de adjuntarlo, debe aparecer como una jar o librería adicional

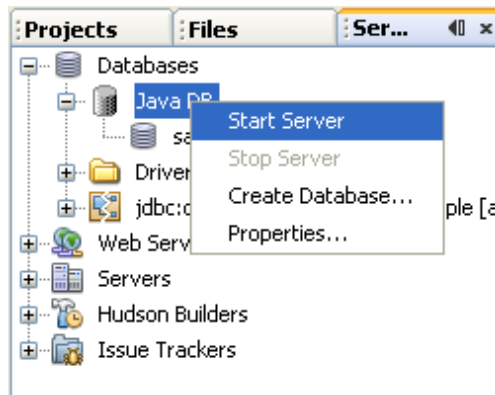




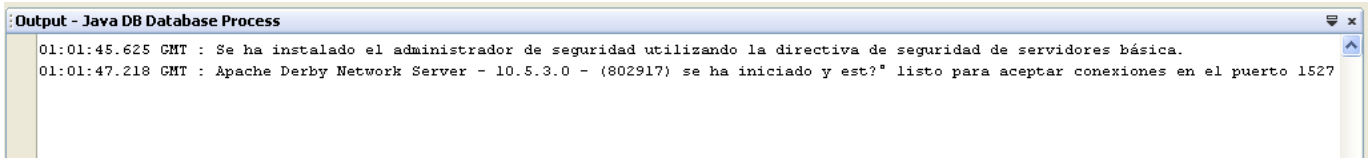
5. A continuación, inicialice el servicio de **Derby**:
- En la pestaña de servicios



- Inicio el servidor de Bases de Datos

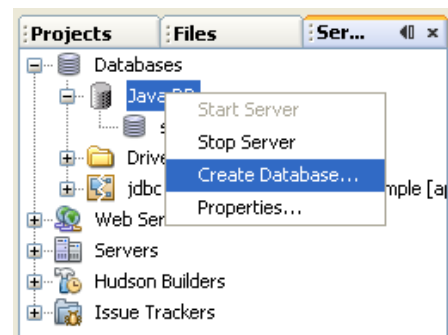


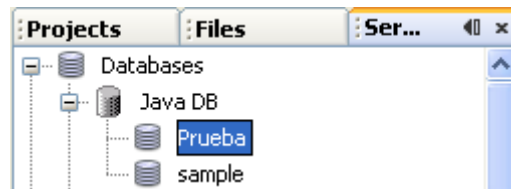
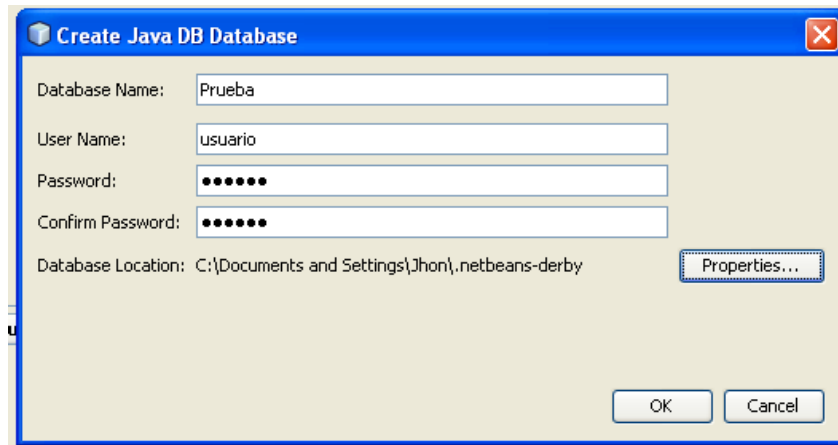
- Tan pronto se inicie el servidor, en la consola aparecerá:



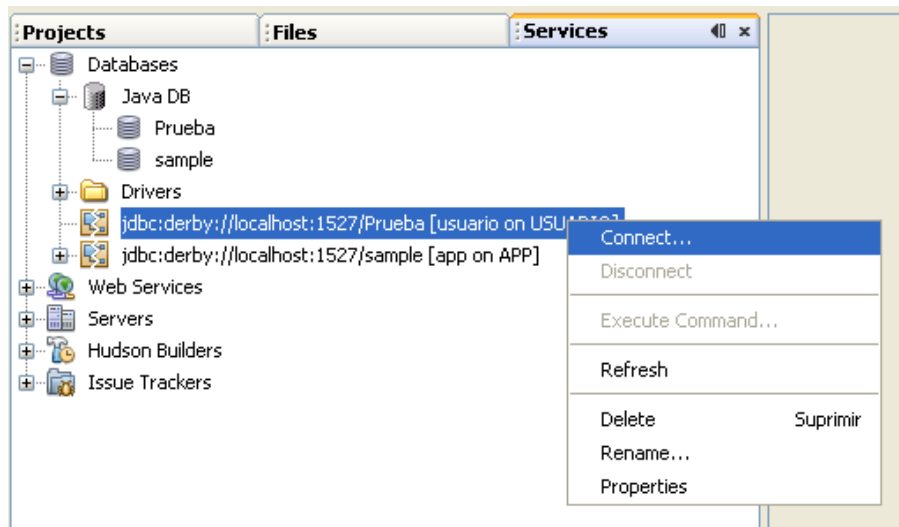
- Ahora cree la Base de Datos

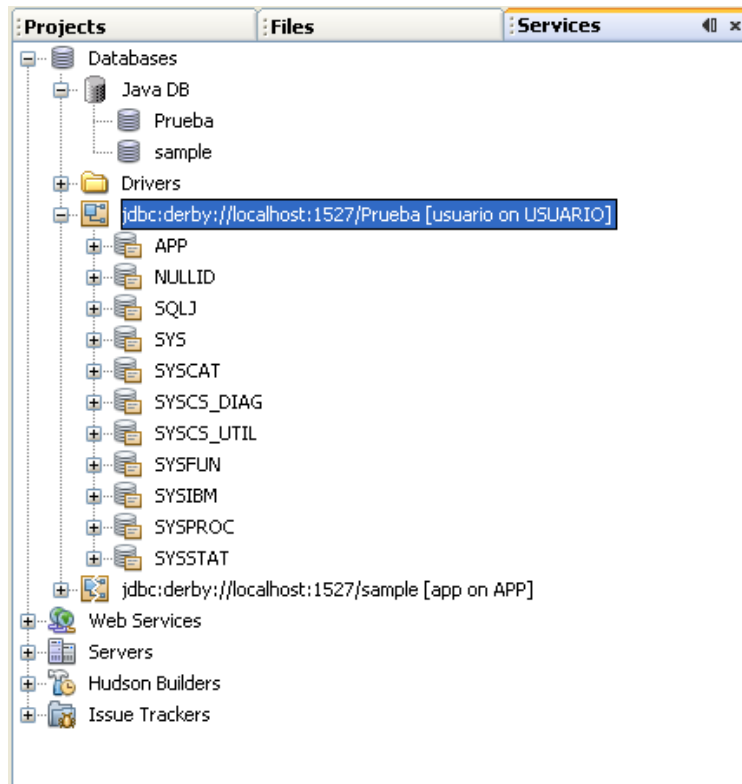
Nombre BD: Prueba
Usuario: usuario
Contraseña: 123456



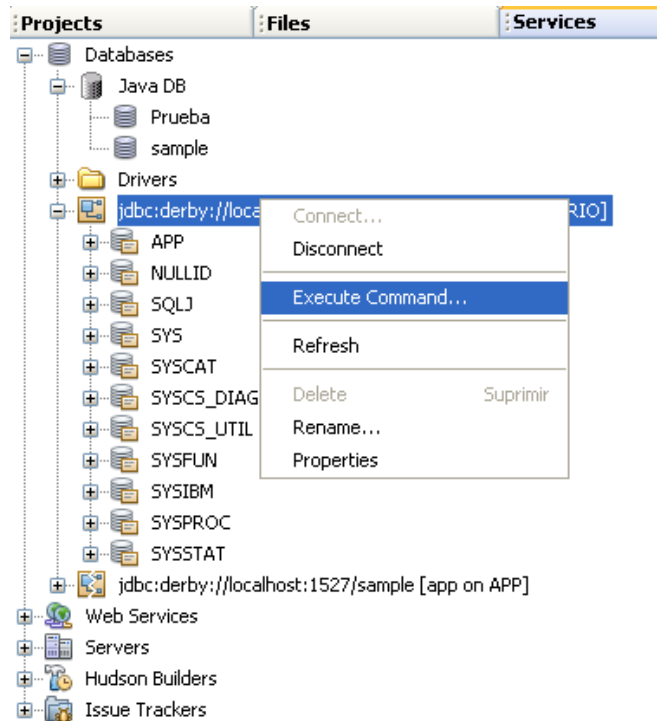


- e. Ahora conecte el servicio: Recuerde que debe conectar el servicio que tiene el nombre del usuario que acabo de crear.





6. A continuación se crea el esquema





SQL Command 1 x
Connection: jdbc:derby://localhost:1527/Prueba [usuario on USUA...]
1 create schema usuario;

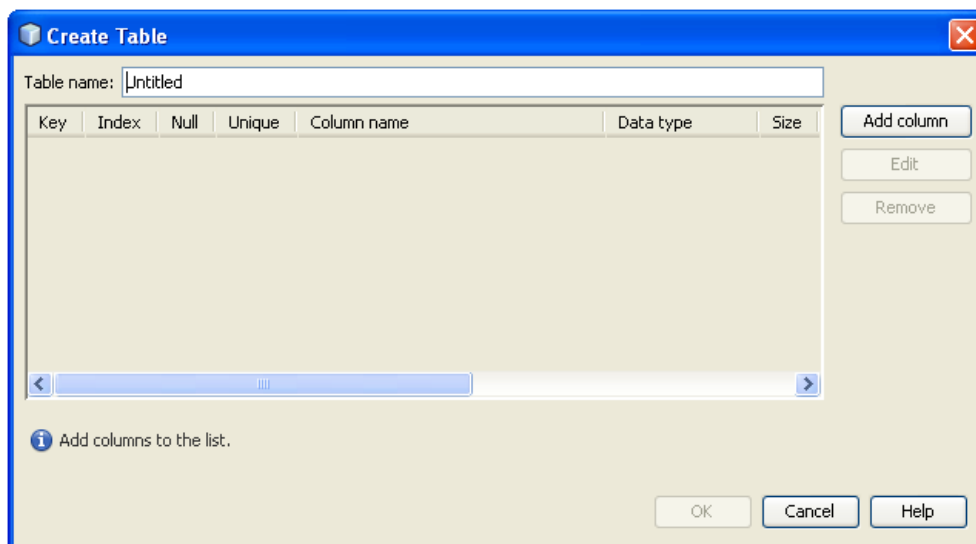
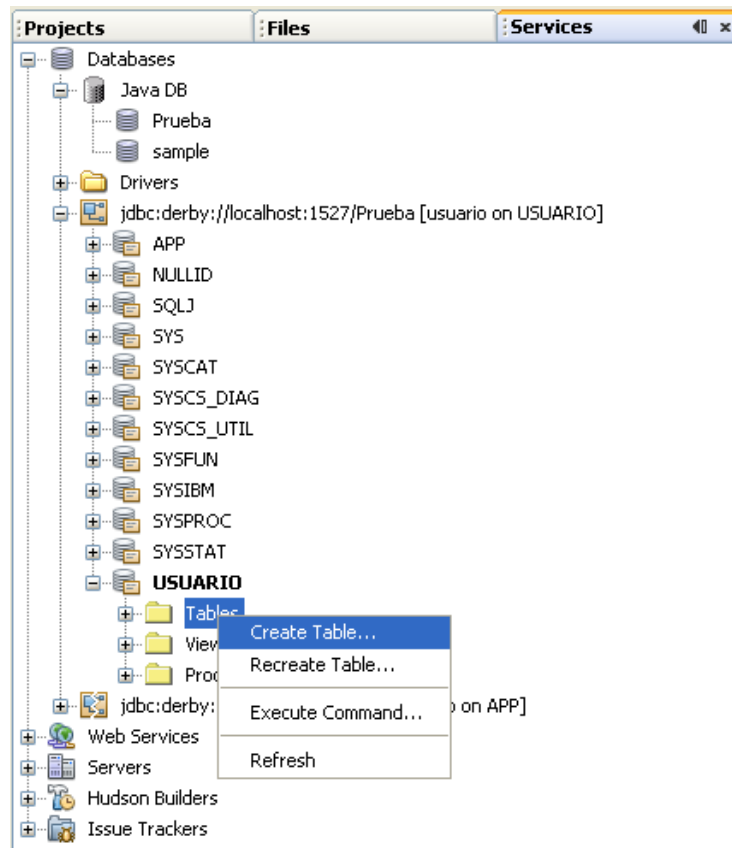
Output
Java DB Database Process x SQL Command 1 execution x
Executed successfully in 0,016 s, 0 rows affected.
Line 1, column 1
Execution finished after 0,016 s, 0 error(s) occurred.

Projects Files Services
Databases
Java DB
Prueba
sample
Drivers
jdbc:derby://localhost:1527/Prueba [usuario on USUARIO]
APP
NULLID
SQLJ
SYS
SYSCAT
SYSCS_DIAG
SYSCS_UTIL
SYSFUN
SYSIBM
SYSPROC
SYSSTAT
USUARIO
Tables
Views
Procedures
jdbc:derby://localhost:1527/sample [app on APP]
Web Services
Servers
Hudson Builders
Issue Trackers



7. Creado el esquema, a continuación cree la primera tabla

Tabla Empleado
Código: Int
Nombre: Varchar
Apellido: Varchar
Edad: Int
Salario: real
Estado: Int





Add Column

Name:

Type:

Size: Scale:

Default:

Constraints

Primary key Unique Null Index

Check:

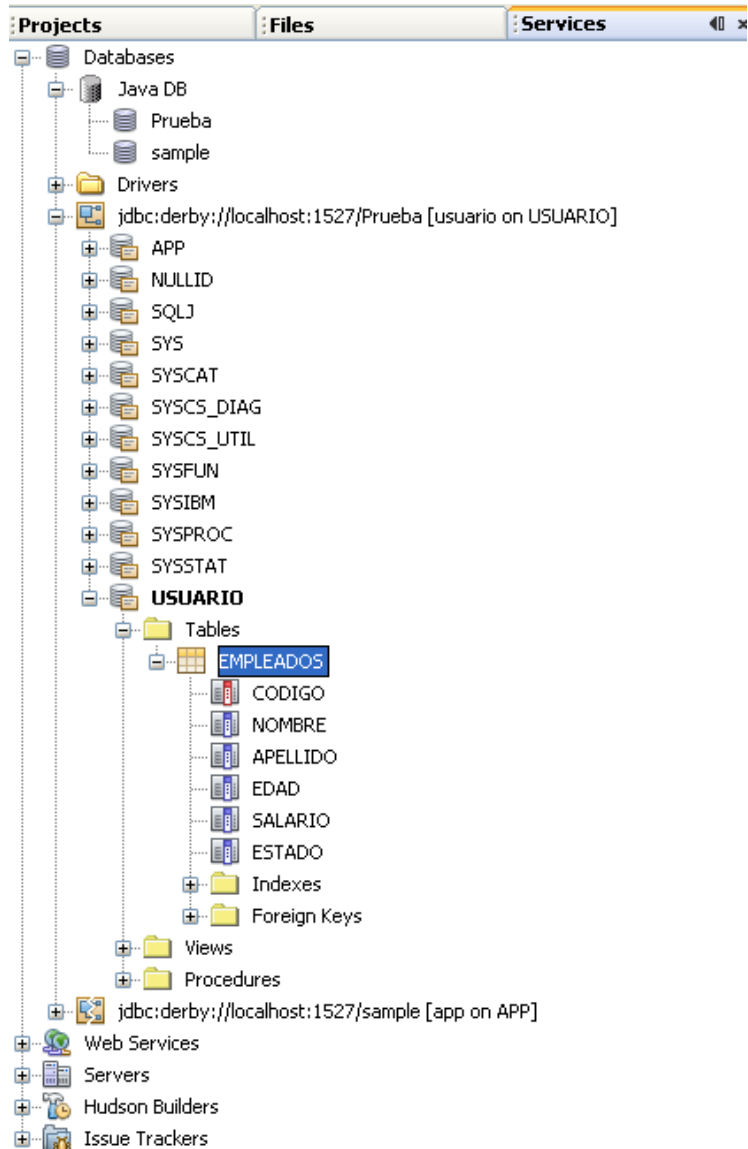
OK Cancel

Create Table

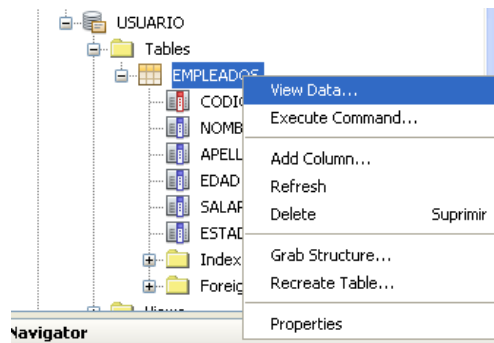
Table name:

| Key | Index | Null | Unique | Column name | Data type | Size |
|-------------------------------------|-------------------------------------|--------------------------|-------------------------------------|-------------|-----------|------|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | codigo | NUMERIC | 0 |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | nombre | VARCHAR | 25 |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | apellido | VARCHAR | 25 |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | edad | NUMERIC | 0 |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | salario | REAL | 0 |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | estado | NUMERIC | 0 |

OK Cancel Help



8. A continuación, inserte datos de prueba





The screenshot shows a SQL Developer window with two tabs: 'SQL Command 1' and 'SQL Command 2'. The connection is 'jdbc:derby://localhost:1527/Ejemplo [user on USER]'. The query in the command window is 'select * from USUARIO.EMPLEADOS'. Below the command window, the results are displayed in a table with the following data:

| # | CODIGO | NOMBRE | APELLIDO | EDAD | SALARIO | ESTADO |
|---|--------|--------|----------|------|----------|--------|
| 1 | 1010 | Pedro | Rojas | 20 | 850000.0 | 1 |
| 2 | 1020 | Maria | Jimenez | 17 | 900000.0 | 1 |

9. Para verificar que los datos fueron ingresados, y que el motor funciona adecuadamente, consulte todos los nombres de los empleados insertados

The screenshot shows a SQL Developer window with three tabs: 'SQL Command 1', 'SQL Command 2', and 'SQL Command 3'. The connection is 'jdbc:derby://localhost:1527/Prueba [usuario on USUA...'. The query in the command window is 'select nombre from USUARIO.EMPLEADOS'. Below the command window, the results are displayed in a table with the following data:

| # | NOMBRE |
|---|--------|
| 1 | Pedro |
| 2 | Maria |



10. Teniendo los datos insertados, a continuación se deben crear las clases que se conectaran y ejecutaran las transacciones. Para este pasa, se debe retornar a la pestaña de "Proyecto"
11. Cree una clase "Conexion"
12. Inserte el siguiente código al interior de la clase

```
public class Conexion {
    private static Connection cn = null;
    private static Driver driver = new org.apache.derby.jdbc.ClientDriver();
    private static String URLBD = "jdbc:derby://localhost:1527/Prueba";
    private static String usuario = "usuario";
    private static String contrasena = "123456";

    public static Connection getConexion() throws SQLException {
        DriverManager.registerDriver(driver);
        cn = DriverManager.getConnection(URLBD, usuario, contrasena);
        return cn;
    }
}
```

13. A continuación se debe crear la clase que opere la conexión, y administre las operaciones DML
14. Cree una Clase Gestor
15. Inserte el siguiente código

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Gestor {

    private Connection con = null;
    private Statement st = null;
    private ResultSet rs = null;

    public Gestor(){
        try {
```

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS
FACULTAD DE INGENIERIA - INGENIERIA DE SISTEMAS
PROGRAMACION AVANZADA



```
//Obtenemos la conexion a Derby
con = Conexion.getConexion();
} catch (SQLException ex) {
    System.out.println("No se pudo realizar la conexion");
}
}

public ResultSet hacerConsulta(){
    String consulta = "SELECT * FROM empleados";
    try {
        //Preparamos la consulta
        st = con.createStatement();
        //Ejecutamos la consulta
        rs = st.executeQuery(consulta);
        mostrarDatos(rs);
    } catch (SQLException ex) {
        System.out.println("No se pudo realizar la consulta");
    }
    return rs;
}

public void mostrarDatos(ResultSet respuesta) throws SQLException{

    System.out.println("CÓDIGO\tNOMBRES\tAPELLO\tEDAD\tSALARIO\tESTADO");
    //Ahora recorreremos el Resultset
    while (respuesta.next()) {
        int codigo, edad, estado;
        String nombres, apellidos;
        double salario;

        //Recogemos los datos llamando al nombre de la columna que el
        //Resultset nos ha devuelto
        codigo = respuesta.getInt("codigo");
        nombres = respuesta.getString("nombre");
        apellidos = respuesta.getString("apellido");
        edad = respuesta.getInt("edad");
        salario = respuesta.getDouble("salario");
        estado = respuesta.getInt("estado");
        //Imprimimos los datos por la consola
        System.out.println(codigo + "\t" + nombres + "\t" + apellidos + "\t"
```



```
        + edad + "\t" + salario + "\t" + estado);
    }
}

public void cerrarCnx() throws SQLException{
    //Cerramos los elementos usados
    rs.close();
    rs = null;
    st.close();
    st= null;
    con.close();
    con=null;
}

public static void main(String[] args) throws SQLException {

    Gestor g=new Gestor();
    g.hacerConsulta();
}
}
```

16. A continuación ejecute la clase “Gestor”, y obtendrá como resultado:

```
run:
CÓDIGO  NOMBRES APELL  EDAD    SALARIO ESTADO
1       Pedro  Reyes  25     2500000.0  1
2       Maria  Castro  19     2700000.0  1
BUILD SUCCESSFUL (total time: 1 second)
```

17. Cree el método que permita insertar un empleado a la tabla.

Si tiene una instalación que no tenga integrado Derby, a continuación se adjunta una página de referencia donde encontrara información de cómo descargar el jar, instalarlo, crear las clases y realizar la ejecución

<http://rchavarria.wordpress.com/2010/06/24/usar-apache-derby-como-base-de-datos/>