Mini Práctica de Laboratorio 2
Empaquetar Acceso a Datos en un JAR
Proyecto Vida Sana

Objetivo

En esta práctica el estudiante aprenderá a organizar el código de acceso a base de datos en una pequeña biblioteca reutilizable. Se creará un proyecto Maven que contendrá la conexión y algunos métodos de acceso a datos (DAO). Este proyecto se empaquetará como un archivo JAR para que pueda ser utilizado por otras aplicaciones.

Contexto

En la práctica anterior se realizó la conexión directa a SQL Server desde una aplicación de consola. Sin embargo, en proyectos reales el código de acceso a datos normalmente se separa en una biblioteca independiente. Esto permite reutilizar la conexión y los métodos de consulta desde diferentes aplicaciones (consola, web, APIs, etc.).

Arquitectura simple utilizada

Aplicación de consola → Biblioteca de acceso a datos (JAR) → SQL Server

Parte 1 – Crear proyecto Maven para acceso a datos

1. Abrir Apache NetBeans
2. File → New Project
3. Java with Maven → Java Application
4. Nombre del proyecto: vida-sana-data
5. Finalizar

(Nota: En este proyecto no vamos a ocupar la clase con el método main)

Parte 2 – Agregar dependencia de SQL Server

Abrir el archivo pom.xml y agregar la dependencia del driver JDBC:
(Sírvase de su práctica anterior para este paso)

```
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>mssql-jdbc</artifactId>
  <version>13.2.1.jre11</version>
</dependency>
```

Parte 3 – Crear clase de conexión

- Crear el paquete: mx.ith.vidasana.data.conexion (Click derecho a su proyecto>New>Java Package)
- Crear la clase ConexionBD (Click derecho a su paquete recién creado del paso anterior New>Java Class)

```
import java.sql.Connection;
import java.sql.DriverManager;

public class ConexionBD {

   public static Connection obtenerConexion() throws Exception {

      String url = "jdbc:sqlserver://nombreServidor;"
            + "databaseName=vida_sana;"
            + "encrypt=true;"
            + "trustServerCertificate=true";

      String usuario = "loginMedico";
      String password = "medico123";

      return DriverManager.getConnection(url, usuario, password);
   }

}
```

Parte 4 – Crear clase DAO

Crear el paquete: mx.ith.vidasana.data.dao
Crear la clase PacienteDAO.

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import mx.edu.vidasana.data.conexion.ConexionBD;

public class PacienteDAO {

   public void listarPacientes(){

      try{
```

```
        Connection conn = ConexionBD.obtenerConexion();
        Statement consulta = conn.createStatement();

        String sql = "SELECT idPaciente, nombre, telefono FROM Pacientes";

        ResultSet rs = consulta.executeQuery(sql);

        while(rs.next()){

           int id = rs.getInt("idPaciente");
           String nombre = rs.getString("nombre");
           String telefono = rs.getString("telefono");

           System.out.println(id + " - " + nombre + " - " + telefono);

        }

        rs.close();
        consulta.close();
        conn.close();

      }catch(Exception e){
        e.printStackTrace();
      }

   }

}
```

Parte 5 – Generar el JAR

Click derecho en el proyecto → Clean and Build.
El archivo JAR se generará dentro de la carpeta:
target/vida-sana-data-1.0.jar
(Este archivo lo ocupará agregar en el paso 7. Verifique la carpeta donde se generó el JAR)

Parte 6 – Crear aplicación cliente

Crear un segundo proyecto Maven llamado vida-sana-app.

Parte 7 – Agregar el JAR al proyecto cliente

1. Click derecho en el proyecto
2. Properties
3. Libraries
4. Add JAR/Folder
5. Seleccionar el JAR generado.

Parte 8 – Usar el DAO

```
import mx.edu.vidasana.data.dao.PacienteDAO;

public class Main {

   public static void main(String[] args) {

      PacienteDAO dao = new PacienteDAO();

      dao.listarPacientes();

   }

}
```

Ahora que sabemos que los métodos siguen funcionando, vamos a formalizar nuestro proyecto "orientando a objetos" nuestras entidades y estableciendo un modelo. Para esto vamos a agregar una clase Paciente para trabajar con él. Siga las indicaciones de su profesor para hacer esto.

Nota:

No tiene que enviar nada, pero considere un JAR así para su proyecto final, porque de esta manera es como va a trabajar.