Maracaibo; 18 de mayo de 2021 — 18/05/2021

CONEXIÓN A BASES DE DATOS

CONTENIDO:

Escribes el código en JAVA para conectarse a una base de datos PostgreSQL. Este Script debe tener los siguientes datos de conexión:

- base de datos = postgres,
- usuario = postgres,
- contraseña = clinora
- puerto de conexión = 5432

Los campos pedidos en la tabla de estudiantes son los siguientes:

- Nombre.
- Apellido.
- Carrera.

Conexion.java (CLASE # 1) → package: modelo

```
1. package modelo;
2. import java.sql.*;
3. public class Conexion {
4.     String url = "jdbc:postgresql://localhost:5432/bd_estudiantes";
```

ESTUDIANTE:

Carlos Daniel Martínez Avila

Firma: Carlos D. Martínez

C.I.: 28090609

```
5.  String user = "postgres", pass = "elimora";
6.  Connection con;
7.  public Connection getConnection() {
8.      try {
9.          Class.forName("org.postgresql.Driver");
10.         con = DriverManager.getConnection(url, user, pass);
11.     } catch (Exception e) {
12.     }
13.     return con;
14. }
15. }
```

Estudiante.java (CLASE #2) → package: modelo

```
1.  package modelo;
2.  public class Estudiante {
3.      int idestudiante;
4.      String nombre;
5.      String apellido;
6.      String carrera;
7.      public Estudiante() {
8.      }
9.      public Estudiante(int idestudiante, String nombre, String apellido, String carrera) {
10.         this.idestudiante
11.         this.nombre = nombre;
12.         this.apellido = apellido;
13.         this.carrera = carrera;
14.     }
15.     public int getId() {
16.         return idestudiante;
17.     }
18.     public void setId(int idestudiante) {
19.         this.idestudiante = idestudiante;
20.     }
21.     public String getNom() {
22.         return nombre;
23.     }
24.     public void setNom(String nombre) {
25.         this.nombre = nombre;
26.     }
27.     public String getAp() {
```

```
28.     return apellido;
29.   }
30.   public void setApellido (String apellido) {
31.     this.apellido = apellido;
32.   }
33.   public String getCarrera () {
34.     return carrera;
35.   }
36.   public void setCarrera (String carrera) {
37.     this.carrera = carrera;
38.   }
39. }
```

EstudianteDAO.java — package: modelo (USAGE #3)

```
1.  package modelo;
2.  import java.sql.Connection;
3.  import java.sql.PreparedStatement;
4.  import java.sql.ResultSet;
5.  import java.util.ArrayList;
6.  import java.util.List;
7.  public class EstudianteDAO {
8.      PreparedStatement ps;
9.      ResultSet rs;
10.     Connection conectar = new Conexion();
11.     Estudiante p = new Estudiante();
12.     Connection con;
13.     public List listar() {
14.         List<Estudiante> datos = new ArrayList<>();
15.         try {
16.             con = conectar.getConnection();
17.             ps = con.prepareStatement("select * from estudiantes");
18.             rs = ps.executeQuery();
19.             while(rs.next()){
20.                 Estudiante p = new Estudiante();
21.                 p.setId(rs.getInt(1));
22.                 p.setNom(rs.getString(2));
23.                 p.setApellido(rs.getString(3));
24.                 p.setCarrera(rs.getString(4));
25.                 datos.add(p);
26.             }
27.         } catch (Exception e) {
```

```
        }
        return datos;
    }
    public int agregar (Estudiante p) {
        int r=0;
        String sql = "insert into estudiantes (idestudiante, nombre, apellido, carrera) values (?,?,?,?)";
        try {
            con = conectar.getConnection();
            ps = con.prepareStatement(sql);
            ps.setInt(1, p.getId());
            ps.setString(2, p.getNom());
            ps.setString(3, p.getAp());
            ps.setString(4, p.getCarrera());
            r = ps.executeUpdate();
            if (r==1) {
                return 1;
            }
            else {
                return 0;
            }
        } catch (Exception e) {
        }
        return r;
    }
    public int Actualizar (Estudiante per) {
        int r=0;
        String sql = "update estudiantes set nombre=?, apellido=?, carrera=? where idestudiante=?";
        try {
            con = conectar.getConnection();
            ps = con.prepareStatement(sql);
            ps.setString(1, per.getNom());
            ps.setString(2, per.getAp());
            ps.setString(3, per.getCarrera());
            ps.setInt(4, per.getId());
            r = ps.executeUpdate();
            if (r==1) {
                return 1;
            }
            else {
                return 0;
            }
        } catch (Exception e) {
        }
    }
```

```
        return r;
    }
    public int Delete (int idestudiante){
        int r=0;
        String sql = "delete from estudiantes where idestudiante = idestudiante;
        try {
            con = conecta.getConnection();
            ps = con.prepareStatement(sql);
            r = ps.executeUpdate();
        } catch (Exception e) {
        }
        return r;
    }
}
```

controlador.java (CLASE # 4) + package

```
1.  package controlador;
2.  import modelo.Estudiante;
3.  import modelo.EstudianteDAO;
4.  import vista.Vista;
5.  import java.awt.event.ActionEvent;
6.  import java.awt.event.ActionListener;
7.  import java.util.List;
8.  import javax.swing.JOptionPane;
9.  import javax.swing.JTable;
10. import javax.swing.SwingConstants;
11. import javax.swing.table.DefaultTableCellRenderer;
12. import javax.swing.table.DefaultTableModel;
13. public class Controlador implements ActionListener {
14.     EstudianteDAO dao = new EstudianteDAO();
15.     Estudiante p = new Estudiante();
16.     Vista vista = new Vista();
17.     DefaultTableModel modelo = new DefaultTableModel();
18.     public Controlador(Vista v) {
19.         this.vista = v;
20.         this.vista.btnListar.addActionListener(this);
21.         this.vista.btnAgregar.addActionListener(this);
22.         this.vista.btnEditar.addActionListener(this);
23.         this.vista.btnDelete.addActionListener(this);
24.         this.vista.btnActualizar.addActionListener(this);
25.         this.vista.btnNuevo.addActionListener(this);
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
```

```
27.     @Override
28.     public void actionPerformed(ActionEvent e) {
29.         if (e.getSource() == vista.btnListar) {
30.             limpiarTabla();
31.             listar(vista.tabla);
32.             nuevo();
33.         }
34.         if (e.getSource() == vista.btnAgregar) {
35.             add();
36.             listar(vista.tabla);
37.             nuevo();
38.         }
39.         if (e.getSource() == vista.btnEditar) {
40.             int fila = vista.tabla.getSelectedRow();
41.             if (fila == -1) {
42.                 JOptionPane.showMessageDialog(vista, "Seleccione una fila");
43.             } else {
44.                 int idestudiante = Integer.parseInt((String) vista.tabla.getValueAt(fila, 0));
45.                 String nom = (String) vista.tabla.getValueAt(fila, 1);
46.                 String apellido = (String) vista.tabla.getValueAt(fila, 2);
47.                 String carrera = (String) vista.tabla.getValueAt(fila, 3);
48.                 vista.txtId.setText("" + idestudiante);
49.                 vista.txtNom.setText(nom);
50.                 vista.txtApe.setText(apellido);
51.                 vista.txtTel.setText(carrera);
52.             }
53.         if (e.getSource() == vista.btnActualizar) {
54.             Actualizar();
55.             listar(vista.tabla);
56.             nuevo();
57.         }
58.         if (e.getSource() == vista.btnDelete) {
59.             delete();
60.             listar(vista.tabla);
61.             nuevo();
62.         }
63.         if (e.getSource() == vista.btnNuevo) {
64.             nuevo();
65.         }
66.     }
67.     void nuevo() {
68.         vista.txtId.setText("");
69.         vista.txtNom.setText("");
```

```
70          vista.txtTel.setText("");
71          vista.txtCorreo.setText("");
72          vista.txtNom.requestFocus();
73      }
74      public void delete(){
75          int fila = vista.tabla.getSelectedRow();
76          if (fila == -1){
77              JOptionPane.showMessageDialog(vista, "Debe seleccionar una fila");
78          } else {
79              int id = Integer.parseInt((String)vista.tabla.getValueAt(fila,0));
80              dao.Delete(id);
81              System.out.println("El resultado es " + id);
82              JOptionPane.showMessageDialog(vista, "Usuario eliminado");
83          }
84          limpiarTabla();
85      }
86      public void Actualizar(){
87          if (vista.txtId.getText().equals("")){
88              JOptionPane.showMessageDialog(vista, "No se identificó el Id");
89          } else {
90              int id = Integer.parseInt(vista.txtId.getText());
91              String nombre = vista.txtNom.getText();
92              String apellido = vista.txtCorreo.getText();
93              String carrera = vista.txtTel.getText();
94              p.setId(id);
95              p.setNom(nombre);
96              p.setApellido(apellido);
97              p.setCarrera(carrera);
98              int r = dao.Actualizar(p);
99              if (r == 1){
100                 JOptionPane.showMessageDialog(vista, "Usuario actualizado");
101             }else{
102                 JOptionPane.showMessageDialog(vista, "Error");
103             }
104         }
105         limpiarTabla();
106     }
107     public void listar(JTable tabla){
108         centrarCeldas(tabla);
```

(7)

```
        modelo = (DefaultTableModel) tabla.getModel();
        tabla.setModel(modelo);
        List<Estudiante> lista = dao.listar();
        Object[] objeto = new Object[4];
        for (int i=0; i<lista.size(); i++){
            objeto[0] = lista.get(i).getId();
            objeto[1] = lista.get(i).getNombre();
            objeto[2] = lista.get(i).getApp();
            objeto[3] = lista.get(i).getCarrera();
            modelo.addRow(objeto);
        }
        tabla.setRowHeight(35);
        tabla.setRowMargin(10);
    }
    void centrarCeldas(JTable tabla){
        DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();
        tcr.setHorizontalAlignment(SwingConstants.CENTER);
        for(int i=0; i<vista.tabla.getColumnCount(); i++){
            tabla.getColumnModel().getColumn(i).setCellRenderer(tcr);
        }
    }
    void limpiarTabla(){
        for(int i=0; i<vista.tabla.getRowCount(); i++){
            modelo.removeRow(i);
            i=(-1);
        }
    }
}
```

ESTRUCTURA GENERAL