	UNIVERSIDAD DON BOSCO ESCUELA DE COMPUTACIÓN
CICLO I	GUIA DE LABORATORIO #6 Programación Orientada a Objetos Interfaces gráficas y JDBC

I.OBJETIVOS

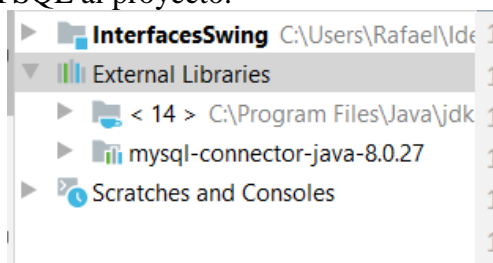
- Que el estudiante aplique sus conocimientos de base de datos.
- Que el estudiante pueda conectar un a aplicación gráfica con una BD.

II. INTRODUCCIÓN

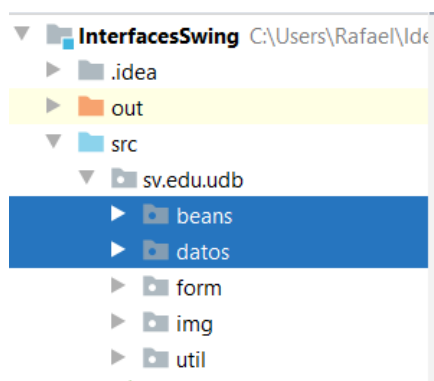
En esta guía continuaremos el uso de Swing y las conexiones a bases de datos. Como antes se había mencionado, IntelliJ IDEA posee excelentes asistentes para el diseño de aplicaciones gráficas, combinado con JDBC se pueden desarrollar aplicaciones potentes para escritorio.

III. PROCEDIMIENTO

1. Abrir el proyecto creado en la sesión anterior llamado “**InterfacesSwing**”, se reutilizará todo lo desarrollado con anterioridad y se le dará una mayor funcionalidad al conectarlo con la base de datos “personabdd” que fue proporcionada en la sesión de JDBC.
2. Agregar el driver de MYSQL al proyecto.



3. Crear los siguientes paquetes:
 - **sv.edu.udb.beans**
 - **sv.edu.udb.datos**



4. En el paquete **sv.edu.udb.util** que fue creado en la sesión anterior se deberá de crear la clase llamada **Conexión** y deberá poseer el siguiente código:

Nota: Tomar en cuenta que el puerto puede ser diferente en su maquina

```
package sv.edu.udb.util;

import java.sql.*;
/**
 * Clase Conexion JDBC
 * @author Rafael Torres
 */
public class Conexion {

    //Valores de conexion a MySql

    private static String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    //El puerto es opcional
    private static String JDBC_URL =
"jdbc:mysql://localhost:3308/personabdd";
    private static String JDBC_USER = "root";
    private static String JDBC_PASS = "";
    private static Driver driver = null;

    //Para que no haya problemas al obtener la conexion de
    //manera concurrente, se usa la palabra synchronized
    public static synchronized Connection getConnection()
        throws SQLException {
        Connection con = null;

        if (driver == null) {
            try {
                //Se registra el driver
                Class.forName(JDBC_DRIVER);

                //con = DriverManager.getConnection (
                //      JDBC_URL, JDBC_USER, JDBC_PASS);
            } catch (Exception e) {
                System.out.println("Fallo en cargar el driver JDBC");
                e.printStackTrace();
            }
        }
        return DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASS);
    }

    //Cierre del resultSet
    public static void close(ResultSet rs) {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
    }

    //Cierre del PreparedStatement
    public static void close(PreparedStatement stmt) {
        try {
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
    }
}
```

```

    }
}

//Cierre de la conexion
public static void close(Connection conn) {
    try {
        if (conn != null) {
            conn.close();
        }
    } catch (SQLException sqle) {
        sqle.printStackTrace();
    }
}
}

```

5. En el paquete llamado **sv.edu.edu.sv** crear la clase llamada **PersonaBeans** y agregar el siguiente código.

```

package sv.edu.udb.beans;

public class PersonaBeans {
    int idPersona;
    String nombrePersona;
    int edadPersona;
    String telefonoPersona;
    String sexoPersona;
    int idOcupacion;
    String fechaNacimiento;

    public PersonaBeans(int idPersona, String nombrePersona, int edadPersona,
String telefonoPersona, String sexoPersona, int idOcupacion, String
fechaNacimiento) {
        this.idPersona = idPersona;
        this.nombrePersona = nombrePersona;
        this.edadPersona = edadPersona;
        this.telefonoPersona = telefonoPersona;
        this.sexoPersona = sexoPersona;
        this.idOcupacion = idOcupacion;
        this.fechaNacimiento = fechaNacimiento;
    }

    public int getIdPersona() {
        return idPersona;
    }

    public void setIdPersona(int idPersona) {
        this.idPersona = idPersona;
    }

    public String getNombrePersona() {
        return nombrePersona;
    }

    public void setNombrePersona(String nombrePersona) {
        this.nombrePersona = nombrePersona;
    }

    public int getEdadPersona() {
        return edadPersona;
    }

    public void setEdadPersona(int edadPersona) {
        this.edadPersona = edadPersona;
    }
}

```

```

}

public String getTelefonoPersona() {
    return telefonoPersona;
}

public void setTelefonoPersona(String telefonoPersona) {
    this.telefonoPersona = telefonoPersona;
}

public String getSexoPersona() {
    return sexoPersona;
}

public void setSexoPersona(String sexoPersona) {
    this.sexoPersona = sexoPersona;
}

public int getIdOcupacion() {
    return idOcupacion;
}


public void setIdOcupacion(int idOcupacion) {
    this.idOcupacion = idOcupacion;
}

public String getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(String fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}
}

```

6. Para los elementos marcados en rojo se deben de agregar y para el elemento marcado en azul requiere una modificación.

 <h2>Datos de Persona</h2>							
Id:	<input type="text"/>						
Ingrese su nombre:	<input type="text"/>						
Ingrese su edad:	<input type="text"/>						
Ingrese su telefono:	<input type="text"/>						
Seleccione su sexo:	Masculino <input type="button" value="v"/>						
Seleccione su ocupación:	<input type="text"/>						
Ingrese su fecha de Nacimiento:	<input type="text"/>						
<input type="button" value="Guardar"/> <input type="button" value="Eliminar"/> <input type="button" value="Limpiar"/>							
<table border="1"> <thead> <tr> <th>Shape</th> <th>Color</th> </tr> </thead> <tbody> <tr> <td>round</td> <td>red</td> </tr> <tr> <td>square</td> <td>green</td> </tr> </tbody> </table>		Shape	Color	round	red	square	green
Shape	Color						
round	red						
square	green						

Estos serían los cambios en las propiedades de los controles

Control	Propiedad	Valor
JLabel1	text	Selelccione su ocupación:
	field name	lblOcupacion
JLabel2	text	Ingrese su fecha de Nacimiento:
	field name	lblFechaNacimiento
btnObtenerDatos	text	Guardar
txtId	editable	false
JComboBox1	model	vacío
	field name	cmbOcupacion
JTextField1	text	vacío
	field name	txtFechaNacimiento
JButton1	text	Eliminar
	field name	btnEliminar

7. En el paquete **sv.edu.udb.datos** crear las clases **PersonaDatos** , **OcupacionesDatos** y agregarles el siguiente código.

PersonaDatos

En esta clase se estarán creando todos los métodos que interactúan con la base de datos, los cuales permitirán realizar Creaciones, Actualizaciones, Eliminaciones y Lecturas.

```
package sv.edu.udb.datos;

import java.sql.*;
import javax.swing.table.DefaultTableModel;
import sv.edu.udb.util.Conexion;
import sv.edu.udb.beans.PersonaBeans;

public class PersonasDatos {
    //Se utiliza un preparedStatement, por lo que podemos
    //utilizar parametros (signos de ?)
    //los cuales posteriormente será sustituidos por el parametro respectivo
    private final String SQL_INSERT
        = "INSERT INTO persona(id_persona,nombre_persona,
edad_persona,telefono_persona,sexo_persona,id_ocupacion,fecha_nac)
VALUES(?,?,?,?,?,?,?)";
    private final String SQL_UPDATE
        = "UPDATE persona SET nombre_persona=?, edad_persona=?,
telefono_persona=?, sexo_persona=?, id_ocupacion=?, fecha_nac=? WHERE
id_persona=?";
    private final String SQL_DELETE
        = "DELETE FROM persona WHERE id_persona = ?";
    private final String SQL_SELECT
        = "SELECT p.id_persona,p.nombre_persona , p.edad_persona,
p.telefono_persona , " +
        "p.sexo_persona, o.ocupacion, p.fecha_nac " +
        "FROM persona p INNER JOIN ocupaciones o ON p.id_ocupacion =
o.id_ocupacion ";
}
```

```

public int insert(PersonaBeans personaBeans) {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null; //no se utiliza en este ejercicio
    int rows = 0; //registros afectados
    try {
        conn = Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_INSERT);
        int index = 1; //contador de columnas
        stmt.setInt(index++, personaBeans.getIdPersona());
        stmt.setString(index++, personaBeans.getNombrePersona());
        stmt.setInt(index++, personaBeans.getEdadPersona());
        stmt.setString(index++, personaBeans.getTelefonoPersona());
        stmt.setString(index++, personaBeans.getSexoPersona());
        stmt.setInt(index++, personaBeans.getIdOcupacion());
        stmt.setString(index, personaBeans.getFechaNacimiento());
        System.out.println("Ejecutando query:" + SQL_INSERT);
        rows = stmt.executeUpdate(); //no. registros afectados
        System.out.println("Registros afectados:" + rows);

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return rows;
}

```

```

public int update(PersonaBeans personaBeans) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_UPDATE);
        int index = 1;
        stmt.setString(index++, personaBeans.getNombrePersona());
        stmt.setInt(index++, personaBeans.getEdadPersona());
        stmt.setString(index++, personaBeans.getTelefonoPersona());
        stmt.setString(index++, personaBeans.getSexoPersona());
        stmt.setInt(index++, personaBeans.getIdOcupacion());
        stmt.setString(index++, personaBeans.getFechaNacimiento());
        stmt.setInt(index, personaBeans.getIdPersona());

        rows = stmt.executeUpdate();
        System.out.println("Registros actualizados:" + rows);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return rows;
}

```

```

public int delete(int idPersona) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_DELETE);
    }
}

```

```

        stmt = conn.prepareStatement(SQL_DELETE);
        stmt.setInt(1, idPersona);
        rows = stmt.executeUpdate();
        System.out.println("Registros eliminados:" + rows);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return rows;
}

/**
 * Metodo que regresa el contenido de la tabla de persona
 */
public DefaultTableModel selectPersona() {
    DefaultTableModel dtm = new DefaultTableModel();
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try {
        conn = Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_SELECT);
        rs = stmt.executeQuery();
        ResultSetMetaData meta = rs.getMetaData();
        int numberOfColumns = meta.getColumnCount();
        //Formando encabezado
        for (int i = 1; i <= numberOfColumns; i++) {
            dtm.addColumn(meta.getColumnLabel(i));
        }
        //Creando las filas para el JTable
        while (rs.next()) {
            Object[] fila = new Object[numberOfColumns];
            for (int i = 0; i < numberOfColumns; i++) {
                fila[i] = rs.getObject(i+1);
            }
            dtm.addRow(fila);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(rs);
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return dtm;
}
}

```

OcupacionesDatos

Para esta clase solo se crearán dos métodos, uno será utilizado para llenar el combobox de cmbOcupacion con los datos de la base de datos y el otro para obtener el id de la ocupación.

```

package sv.edu.udb.datos;

import sv.edu.udb.util.Conexion;

import javax.swing.table.DefaultTableModel;
import javax.swing.DefaultComboBoxModel;

```

```

import java.sql.*;

public class OcupacionesDatos {

    private final String SQL_SELECT
        = "SELECT id_ocupacion,ocupacion FROM ocupaciones ORDER BY
id_ocupacion";

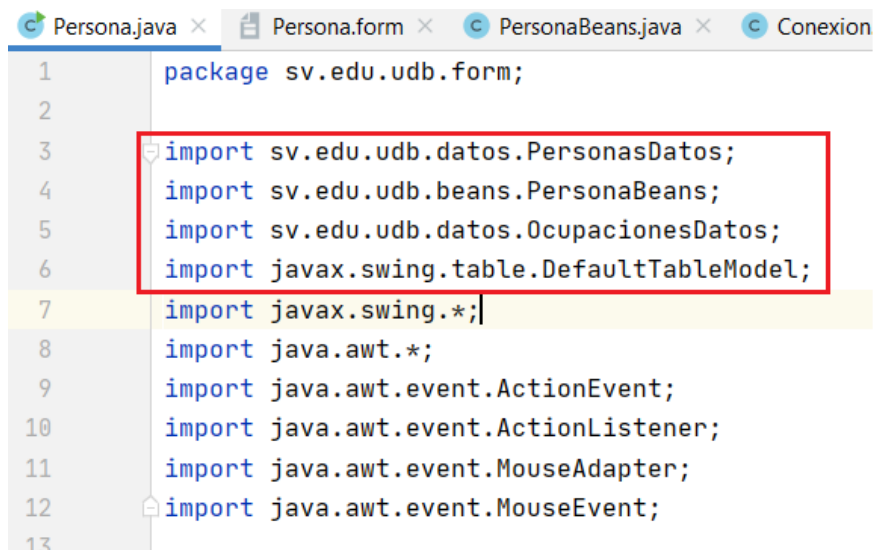
    private final String SQL_SELECT_IDOCUPACION
        = "SELECT id_ocupacion FROM ocupaciones where ocupacion =?";

    public DefaultComboBoxModel selectOcupaciones(){
        DefaultComboBoxModel dtm = new DefaultComboBoxModel();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            conn = Conexion.getConnection();
            stmt = conn.prepareStatement(SQL_SELECT);
            rs = stmt.executeQuery();
            //Creando las filas para el JTable
            while (rs.next()) {
                dtm.addElement(rs.getObject(2));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            Conexion.close(rs);
            Conexion.close(stmt);
            Conexion.close(conn);
        }
        return dtm;
    }

    public int getIdOcupacion(String ocupacion){
        int idOcupacion = 0;
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        System.out.println("ocupacion " + ocupacion);
        System.out.println("ocupacion " + SQL_SELECT_IDOCUPACION);
        try {
            conn = Conexion.getConnection();
            stmt = conn.prepareStatement(SQL_SELECT_IDOCUPACION);
            stmt.setString(1, ocupacion);
            rs = stmt.executeQuery();
            //Creando las filas para el JTable
            while (rs.next()) {
                idOcupacion = rs.getInt(1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            Conexion.close(rs);
            Conexion.close(stmt);
            Conexion.close(conn);
        }
        return idOcupacion;
    }
}

```


8. Para este punto ya se ha creado la mayoría de la lógica de la aplicación ahora solo falta invocarla desde la vista, se deberán de importar las siguientes clases en **Persona.java**



```
1 package sv.edu.udb.form;
2
3 import sv.edu.udb.datos.PersonasDatos;
4 import sv.edu.udb.beans.PersonaBeans;
5 import sv.edu.udb.datos.OcupacionesDatos;
6 import javax.swing.table.DefaultTableModel;
7 import javax.swing.*;
8 import java.awt.*;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.awt.event.MouseAdapter;
12 import java.awt.event.MouseEvent;
```

9. Se deberán de instancias los objetos marcados en rojo en la siguiente imagen.

```
public class Persona extends JFrame{
    private JTextField txtId;
    private JTextField txtNombre;
    private JTextField txtEdad;
    private JTextField txtTelefono;
    private JComboBox cmbSexo;
    private JButton btnObtenerDatos;
    private JButton btnLimpiar;
    private JLabel lblNombre;
    private JLabel lblId;
    private JPanel pnlPersona;
    private JTable tblDatos;
    private JButton eliminarButton;
    private JLabel lblEdad;
    private JLabel lblTelefono;
    private JLabel lblSexo;
    private JLabel lblOcupacion;
    private JLabel lblFechaNacimiento;
    private JComboBox cmbOcupacion;
    private JTextField txtFechaNacimiento;

    DefaultTableModel modelo=null;
    PersonaBeans personaBeans = null;
    PersonasDatos personaDatos = new PersonasDatos();
    OcupacionesDatos ocupacionesDatos = new OcupacionesDatos();
}
```

10. En el constructor se realizarán varios cambios, entre los cuales está el tamaño del JFrame, adicional se comentará el código marcado en azul el cual permitir inicializar la tabla con valores fijo y ahora se agregó el código marcado en rojo el cual permite obtener los datos de la base datos y guardarlos en un modelo que es pasado a la tabla, además de agregar el modelo al combobox de ocupación, el cual también es obtenido de la base de datos.

```
public Persona(String title) {
    super(title);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setContentPane(pnlPersona);
    this.setMinimumSize(new Dimension( width: 750, height: 600));
    this.setLocationRelativeTo(getParent());

    //Arreglo de objeto, para inicializar con vacio la tabla
    //Object [][] data=null;
    //Arreglo de String para crear los nombres de las columnas
    /*String[] columns= {
        "Id", "Nombres", "Edad", "Telefono", "Sexo"
    };*/
    //Instancia del modelo
    //modelo= new DefaultTableModel(data, columns);
    //Seteo del modelo, el cual tendra la estructura que permitira
    //a la tabla representar los datos
    //tblDatos.setModel(modelo);

    modelo=personaDatos.selectPersona();
    tblDatos.setModel(modelo);
    cmbOcupacion.setModel(ocupacionesDatos.selectOcupaciones());
}
```

11. El método **btnObtenerDatos**, deberá ser modificado con el siguiente código.

```
12. private void btnObtenerDatos() {
    int id;
    String nombres;
    int edad;
    String telefono;
    String sexo;
    int idOcupacion;
    String fechaNacimiento;

    if (txtId.getText().isEmpty()) {
        id = 0;
    } else {
        id = Integer.parseInt(txtId.getText());
    }

    nombres = txtNombre.getText();
    edad = Integer.parseInt(txtEdad.getText());
    telefono = txtTelefono.getText();
    sexo = cmbSexo.getSelectedItem().toString();
    idOcupacion =
    ocupacionesDatos.getIdOcupacion(cmbOcupacion.getSelectedItem().toString());
    fechaNacimiento = txtFechaNacimiento.getText();

    personaBeans = new
    PersonaBeans(id,nombres,edad,telefono,sexo,idOcupacion,fechaNacimiento);

    if(btnObtenerDatos.getText().equals("Guardar")) {
        personaDatos.insert(personaBeans);
    } else if (btnObtenerDatos.getText().equals("Editar")) {
        personaDatos.update(personaBeans);
    }
    modelo=personaDatos.selectPersona();
    tblDatos.setModel(modelo);
}
}
```

12. El método de **btnLimpiar**, deberá ser modificado con el siguiente código.

```
private void btnLimpiar() {  
    txtId.setText("");  
    txtNombre.setText("");  
    txtEdad.setText("");  
    txtTelefono.setText("");  
    cmbSexo.setSelectedIndex(0);  
    cmbOcupacion.setSelectedIndex(0);  
    txtFechaNacimiento.setText("");  
    btnObtenerDatos.setText("Guardar");  
}
```

13. El método **tblObtenerData**, deberá ser modificado con el siguiente código.

```
private void tblObtenerData(MouseEvent e) {  
    int fila = tblDatos.rowAtPoint(e.getPoint());  
    int columna = tblDatos.columnAtPoint(e.getPoint());  
  
    if ((fila > -1) && (columna > -1)){  
        txtId.setText(modelo.getValueAt(fila,0).toString());  
        txtNombre.setText(modelo.getValueAt(fila,1).toString());  
        txtEdad.setText(modelo.getValueAt(fila,2).toString());  
        txtTelefono.setText(modelo.getValueAt(fila,3).toString());  
        cmbSexo.setSelectedItem(modelo.getValueAt(fila,4).toString());  
        cmbOcupacion.setSelectedItem(modelo.getValueAt(fila,5).toString());  
        txtFechaNacimiento.setText(modelo.getValueAt(fila,6).toString());  
        btnObtenerDatos.setText("Editar");  
    }  
}
```

14. Al botón **btnEliminar** se le deberá de crear un Listener de tipo **ActionPerformed** el cual deberá de tener la invocación del método **btnEliminarDatos** y que contendrá el código mostrado en la imagen.

```
88      btnEliminar.addActionListener(new ActionListener() {  
89          @Override  
90          public void actionPerformed(ActionEvent e) {  
91              btnEliminarDatos();  
92          }  
93      });  
94  }  
95  
96  private void btnEliminarDatos() {  
97  
98      personaDatos.delete(Integer.parseInt(txtId.getText()));  
99      btnLimpiar();  
100     modelo=personaDatos.selectPersona();  
101     tblDatos.setModel(modelo);  
102 }  
103
```

15. Al ejecutar el programa este aparecerá con la información de la base de datos cargada en la tabla, así como el combobox lleno de las ocupaciones

 Ingreso de Datos

Datos de Persona

Id:

Ingrese su nombre:

Ingrese su edad:

Ingrese su telefono:

Seleccione su sexo: Masculino

Seleccione su ocupación: Doctor

Ingrese su fecha de Nacimiento:

Guardar Eliminar Limpiar

id_persona	nombre_p...	edad_pers...	telefono_p...	sexo_pers...	ocupacion	fecha_nac
2	Alejandro ...	45	7722-4455	Masculino	Doctor	1999-01-05
16	Emerson ...	48	7895-6589	Masculino	Doctor	1999-08-03
17	Helen Plat...	45	7722-2222	Femenino	Doctor	1999-08-03
12	Fernando ...	21	7667-7890	Masculino	Emprende...	2001-05-07
47	Laura Per...	30	78958-586	Femenino	Emprende...	1992-08-10
3	Rafael To...	58	2292-6599	Masculino	Profesor	1986-02-19
15	Emerson ...	22	7123-9800	Masculino	Profesor	1999-08-03
18	Kenia Ort...	30	7789-5522	Femenino	Profesor	1991-02-15

Si se da click sobre un registro, este llenara de manera automática el formulario, cambiando incluso el texto del botón **btnObtenerDatos** que tiene **Guardar**, por el texto **Editar**, si se modifica algún dato este será actualizado en la base de datos y también mostrado en la tabla.

 Ingreso de Datos

Datos de Persona

Id:

Ingrese su nombre:

Ingrese su edad:

Ingrese su telefono:

Seleccione su sexo: Femenino

Seleccione su ocupación: Emprendedor

Ingrese su fecha de Nacimiento:

Editar Eliminar Limpiar

id_persona	nombre_p...	edad_pers...	telefono_p...	sexo_pers...	ocupacion	fecha_nac
2	Alejandro ...	45	7722-4455	Masculino	Doctor	1999-01-05
16	Emerson ...	48	7895-6589	Masculino	Doctor	1999-08-03
17	Helen Plat...	45	7722-2222	Femenino	Doctor	1999-08-03
12	Fernando ...	21	7667-7890	Masculino	Emprende...	2001-05-07
47	Laura Per...	30	78958-586	Femenino	Emprende...	1992-08-10
3	Rafael To...	58	2292-6599	Masculino	Profesor	1986-02-19
15	Emerson ...	22	7123-9800	Masculino	Profesor	1999-08-03
18	Kenia Ort...	30	7789-5522	Femenino	Profesor	1991-02-15

Si se ingresa un nuevo registro, se puede observar que no se le ingresa el ID, ya que este campo es auto incrementable en la base de datos, el botón estará con el texto Guardar y al darle click se insertará y actualizará la tabla.

The screenshot shows a web application window titled "Ingreso de Datos". Inside, there's a section titled "Datos de Persona" with a form for entering personal data. The form fields are:

- Id:** (empty text box)
- Ingrese su nombre:** (text box containing "Yoselin Ramirez")
- Ingrese su edad:** (text box containing "26")
- Ingrese su telefono:** (text box containing "7898-7859")
- Seleccione su sexo:** (dropdown menu showing "Femenino")
- Seleccione su ocupación:** (dropdown menu showing "Emprendedor")
- Ingrese su fecha de Nacimiento:** (text box containing "1995-03-04")

Below the form are three buttons: "Guardar", "Eliminar", and "Limpiar". At the bottom, there is a table with the following data:

id_persona	nombre_p.	edad_per.	telefono_p.	sexo_per...	ocupacion	fecha_nac
16	Emerson ...	48	7895-6589	Masculino	Doctor	1999-08-03
17	Helen Pla...	45	7722-2222	Femenino	Doctor	1999-08-03
12	Fernando...	21	7667-7890	Masculino	Emprend...	2001-05-07
47	Laura Per...	35	78958-586	Femenino	Emprend...	1992-08-10
48	Yoselin R...	26	7898-7859	Femenino	Emprend...	1995-03-04
3	Rafael To...	58	2292-6599	Masculino	Profesor	1986-02-19
15	Emerson ...	22	7123-9800	Masculino	Profesor	1999-08-03
18	Kenia Ort...	30	7789-5522	Femenino	Profesor	1991-02-15

Para eliminar un registro, primero se deberá de seleccionar un valor de la tabla y ya con el formulario lleno, se deberá dar click en Eliminar lo cual provocará que se elimine de la base de datos y se actualice la tabla.

Si damos click en limpiar restaurará todos los valores de los campos de texto.

IV. EJERCICIOS COMPLEMENTARIOS

- Crear un mantenimiento para las tablas
 - Alumno
 - Materias
 - AlumnoMateria

Esta ya se había realizado con anterioridad, solo que ahora deberá utilizar formularios para interactuar con el usuario.