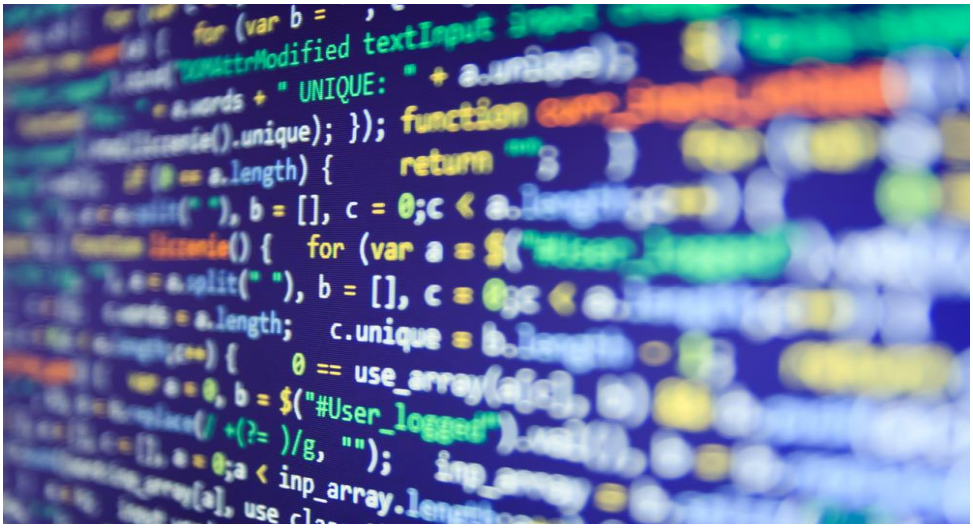
SEP | TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CD.GUZMAN

Ingeniería en Sistemas Computacionales.



ALUMNOS:

Miguel Diaz Mejía.

Ángel Diaz Mejía.

Rubén Esaú García Gómez.

PROFESORA: María Rosario Montes Álvarez.

MATERIA: Ingeniería de software.

TRABAJO: Manual de Programador DSMAFORNU.

Indice:

Introducion

El siguiente documento tiene como objetivo brindrarle al personal encargado de darle mantenimiento o mejoras al sistema la facilidad de entender y comprender las funciones que se realizan mediante la utilización de métodos, explicando de manera de texto lo que se realiza en cada función de los modulos con los que cuenta el sistema esto se realiza de dicha manera para cuando se llegue al visualizar el código se este mas familiarizado con el mismo.

Diagramas UML.

En la figura 1 se muestra el diagrama del caso de uso general.

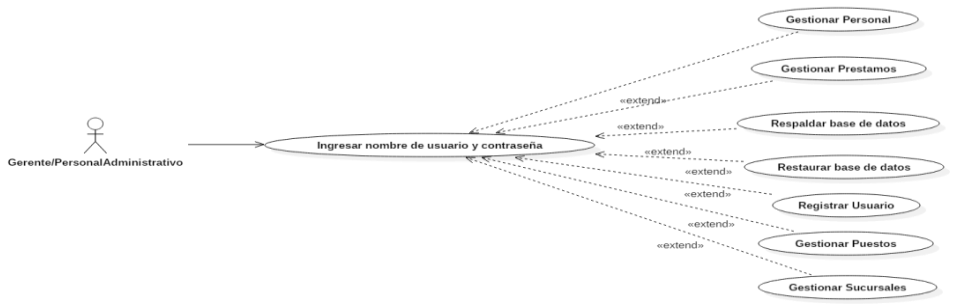


Figura 1. Diagrama del caso de uso general.

En la figura 2 se muestra el diagrama del caso de uso Gestionar Personal.

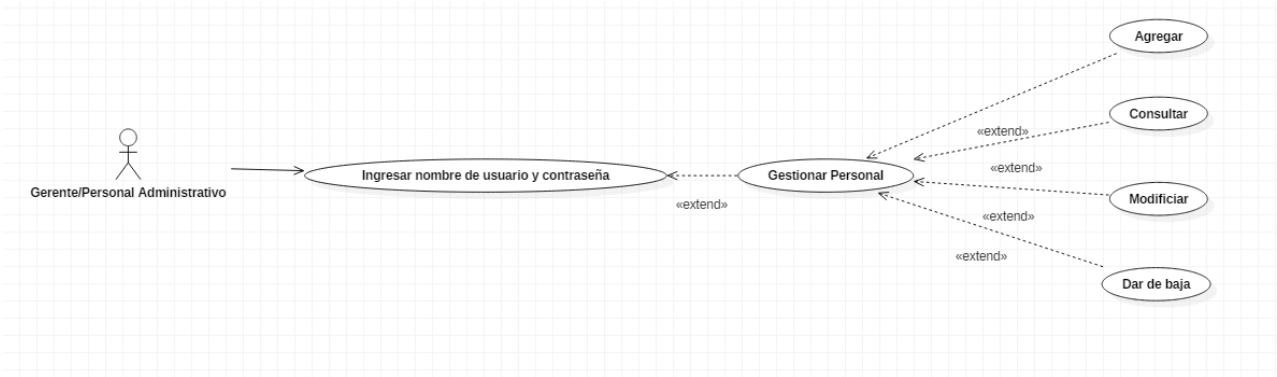


Figura 2. Diagrama del caso de uso Gestionar Personal.

En la figura 3 se muestra el caso de uso de Gestionar Préstamos.

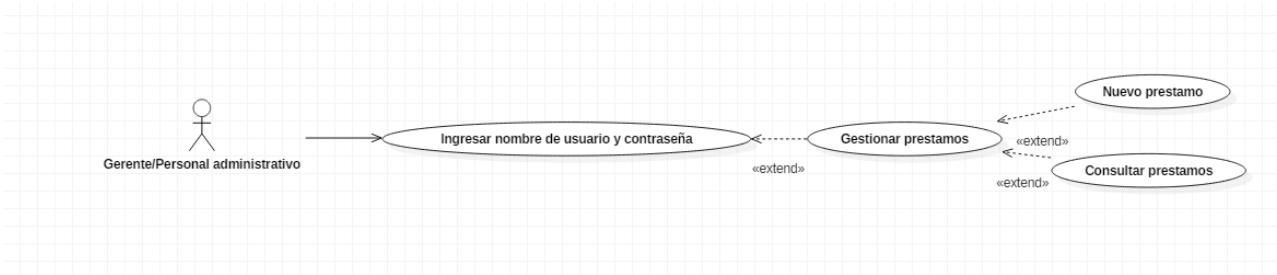


Figura 3. Caso de uso de Gestionar Préstamos.

En la figura 4 se muestra el diagrama de caso de uso Gestionar Puestos.

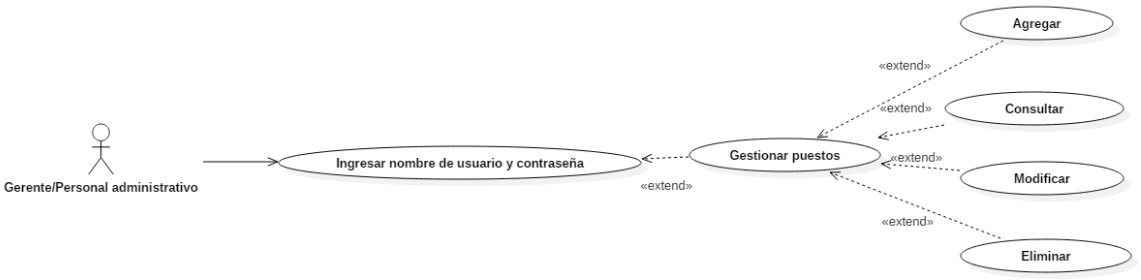


Figura 4. Diagrama del caso de uso Gestionar Puestos.

En la figura 5 se muestra el caso de uso de Gestionar Sucursales.

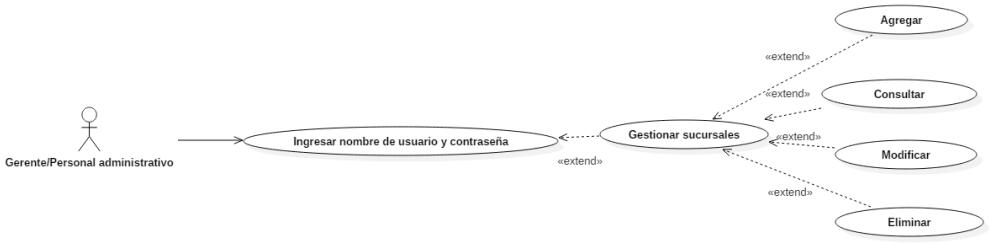


Figura 5. Caso de uso de Gestionar Sucursales.

En la figura 6 se muestra el caso de uso de Registrar Usuario.

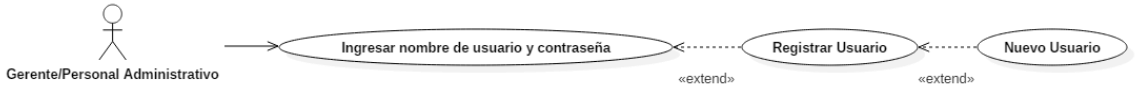


Figura 6. Caso de uso de Registrar Usuario.

Diagrama de Clases.

En la figura 7 se muestra el diagrama de clases del sistema.

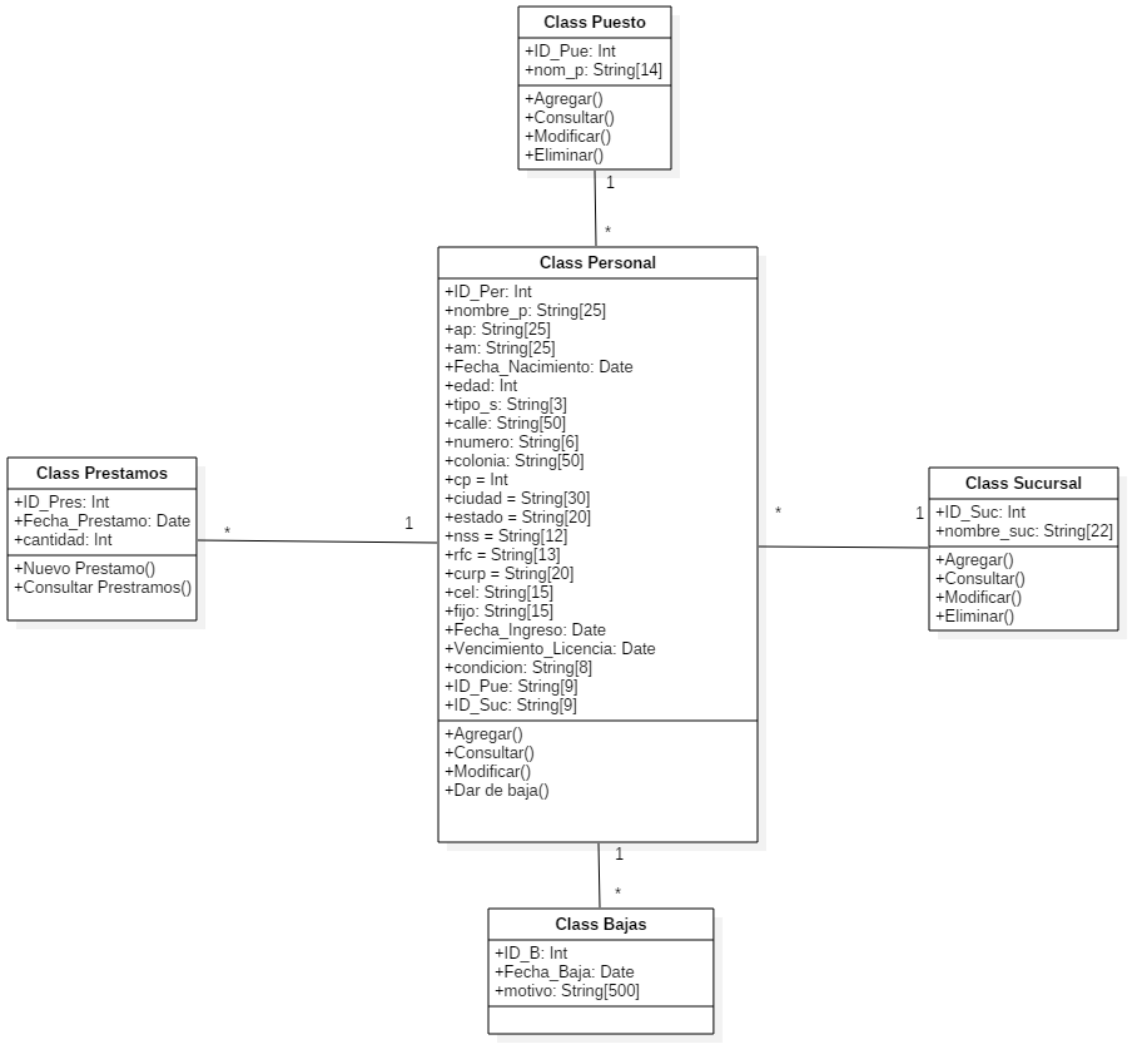


Figura 7. Diagrama de clases del sistema.

Diagramas de Componentes.

En esta subsección se describen de manera gráfica los elementos físicos del sistema, sus relaciones y las funciones con las que cuenta.

En la figura 8 se muestra el diagrama de componentes del proyecto.

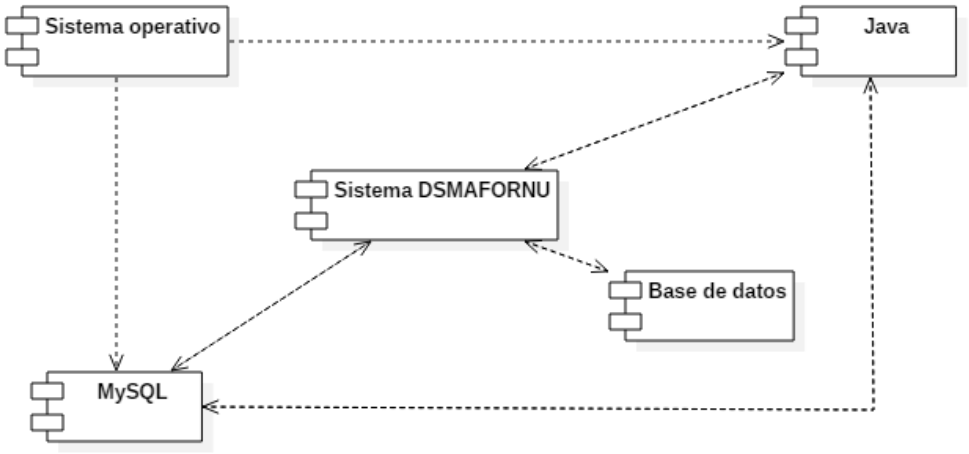


Figura 8. Diagrama de Componentes del proyecto.

En la figura 9 se muestra el diagrama de componentes del sistema.

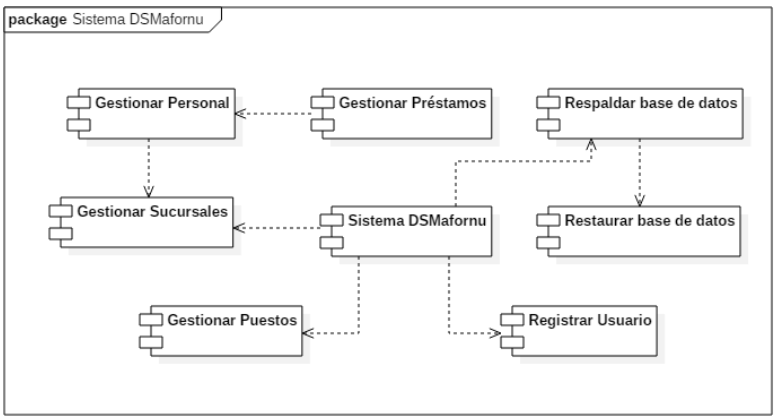


Figura 9. Diagrama de Componentes del sistema.

En la figura 10 se muestra el diagrama de componentes de Gestionar Personal.

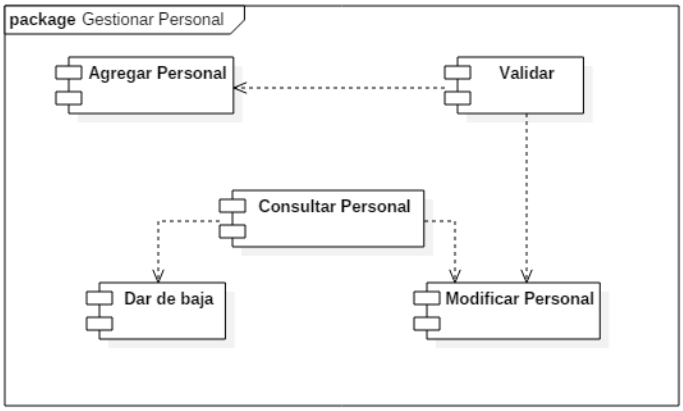


Figura 10. Diagrama de Componentes de Gestionar Personal.

En la figura 11 se muestra el diagrama de componentes de Gestionar Préstamos.

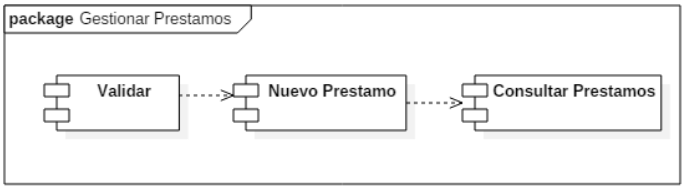


Figura 11. Diagrama de Componentes de Gestionar Préstamos.

En la figura 12 se muestra el diagrama de componentes de Gestionar Puestos.

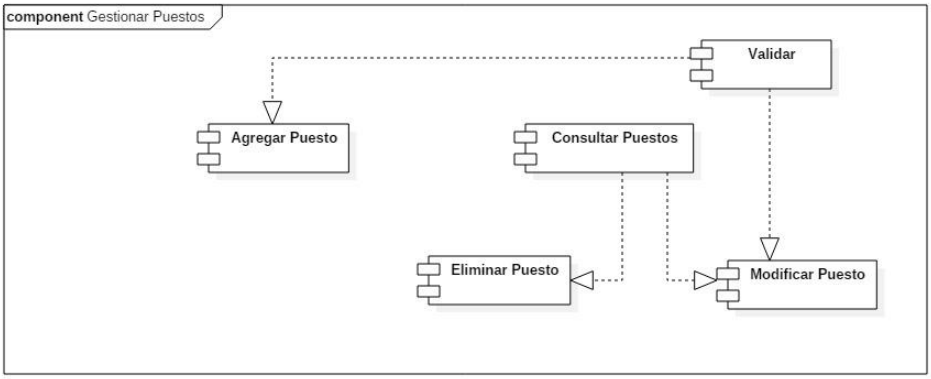


Figura 12. Diagrama de Componentes de Gestionar Puestos.

En la figura 13 se muestra el diagrama de componentes de Gestionar Sucursales.

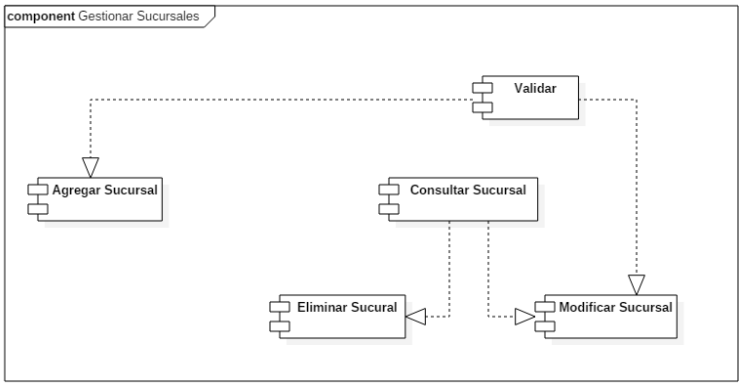


Figura 13. Diagrama de Componentes de Gestionar Sucursales.

Diagrama de Despliegue.

En esta subsección se muestra la disposición física de los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos.

En la figura 14 se muestra el diagrama de despliegue del proyecto.

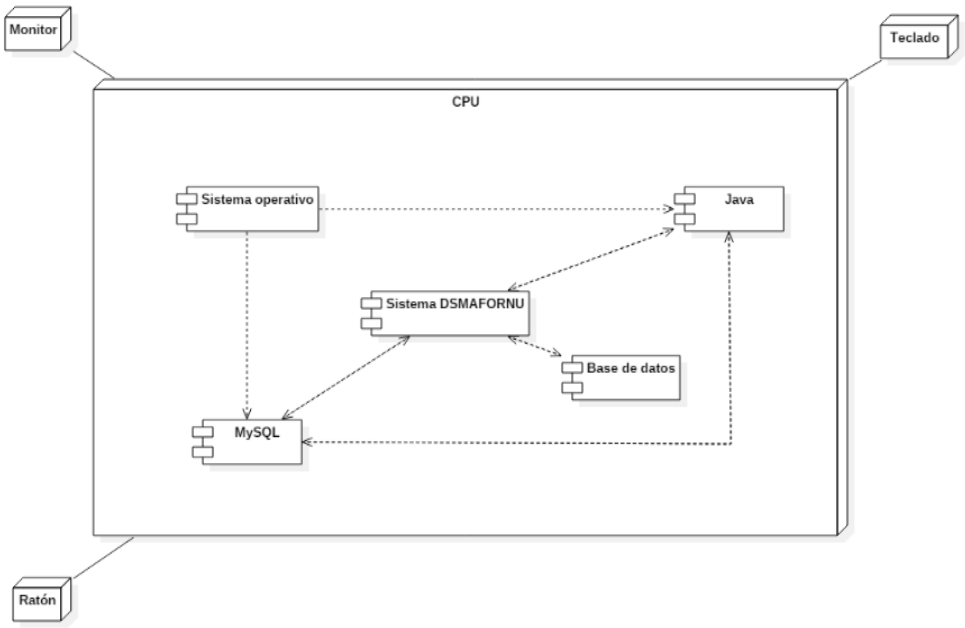


Figura 14. Diagrama de Despliegue del proyecto.

Diagrama Entidad – Relación.

En la figura 15 se muestra el diagrama E-R.

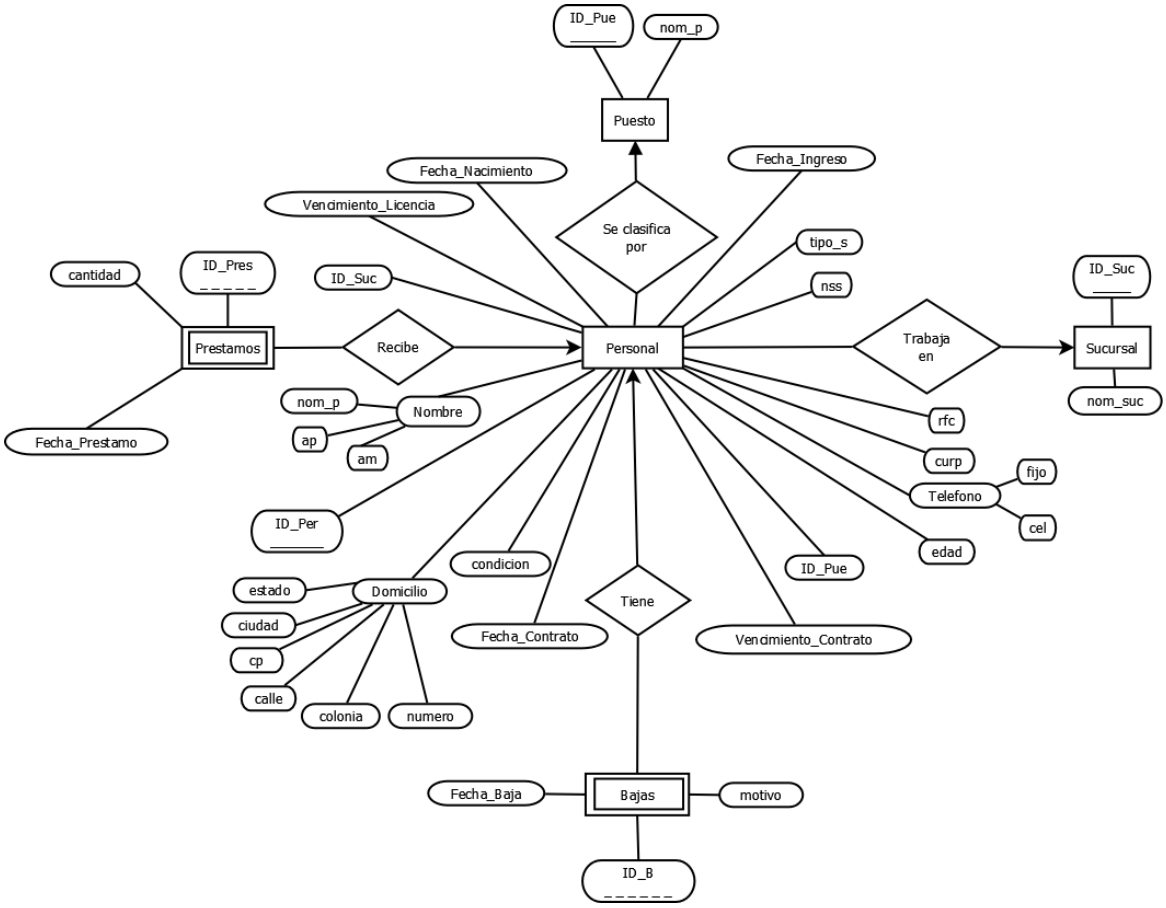


Figura 15. Diagrama E-R

Diccionario de datos.

En esta subsección se muestran todos los atributos con los que cuenta el sistema junto con su descripción y dominio.

En la tabla 1 se muestra el diccionario de datos de los atributos del diagrama entidad-relación.

Atributo	Descripción	Dominio
Am	Almacena el apellido materno de cada personal.	Conjunto de cadenas válidas para el atributo am compuesta de 25 caracteres de la A-Z con espacios, puntos y acentos.
Ap	Almacena el apellido paterno de cada personal.	Conjunto de cadenas válidas para el atributo ap compuesta de 25 caracteres de la A-Z con espacios, puntos y acentos.
calle	Almacena la calle donde vive cada personal.	Conjunto de cadenas válidas para el atributo am compuesta de 50 caracteres de la A-Z con espacios, puntos y acentos.
cantidad	Almacena la cantidad a abonar cada semana de un préstamo.	Valor numérico entero positivo en el rango de 1-9999 para el atributo cantidad.
cel	Almacena el número de celular de cada personal.	Cadena válida para el atributo cel compuesta por caracteres enteros del 0-9 con un rango de 15 caracteres.
ciudad	Almacena la ciudad donde vive cada personal.	Conjunto de cadenas válidas para el atributo ciudad compuesta de 30 caracteres de la A-Z con espacios, puntos y acentos.
colonia	Almacena la colonia donde vive cada personal.	Conjunto de cadenas válidas para el atributo colonia compuesta de 50 caracteres de la A-Z con espacios, puntos y acentos.
condicion	Almacena si el personal está activo o inactivo.	Valor booleano true-false para el atributo condición.
cp	Almacena el número de código postal donde vive cada personal.	Valor numérico entero positivo en el rango de 0-99999 para el atributo cp.
curp	Almacena el curp de cada personal.	Conjunto de cadenas válidas para el atributo curp compuesta de 20 caracteres de la A-Z con espacios, puntos y acentos.

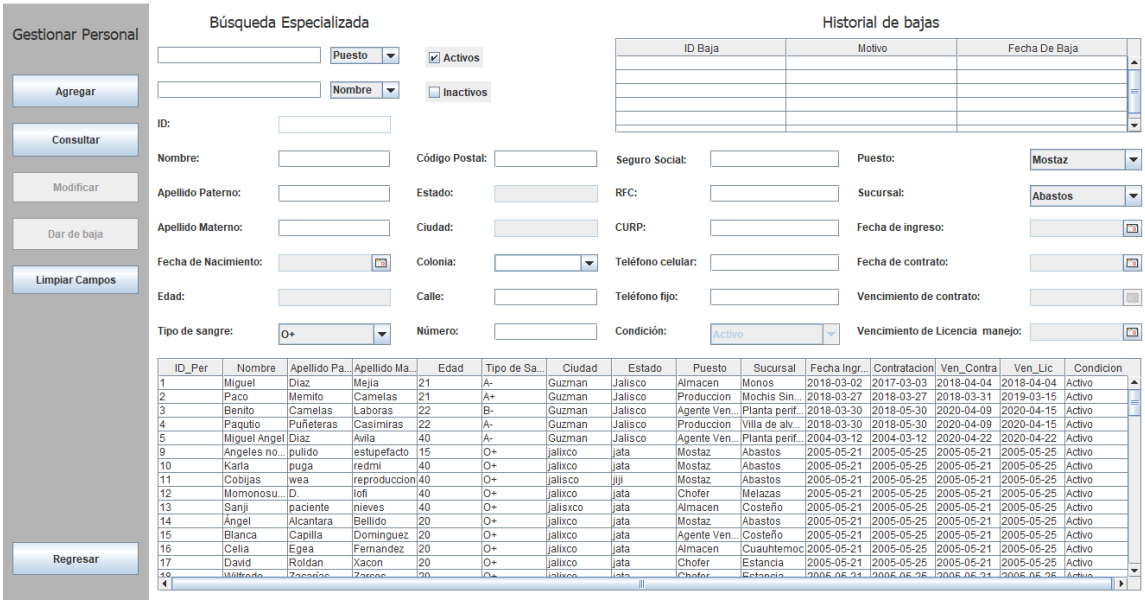
edad	Almacena la edad de cada personal.	Valor numérico entero positivo en el rango de 1-99 para el atributo cantidad.
estado	Almacena el estado donde vive cada personal.	Conjunto de cadenas válidas para el atributo estado compuesta de 20 caracteres de la A-Z con espacios, puntos y acentos.
Fecha_Baja	Almacena la fecha cuando se da de baja a un personal.	Conjunto de cadenas válidas tipo Date para el atributo Fecha_Baja, con un formato dd/mm/aaa.
Fecha_Contrato	Almacena la fecha del inicio de contrato de cada personal.	Conjunto de cadenas válidas tipo Date para el atributo Fecha_Contrato, con un formato dd/mm/aaa.
Fecha_Ingreso	Almacena la fecha de ingreso de cada personal.	Conjunto de cadenas válidas tipo Date para el atributo Fecha_Ingreso, con un formato dd/mm/aaa.
Fecha_Nacimiento	Almacena la fecha de nacimiento de cada personal.	Conjunto de cadenas válidas tipo Date para el atributo Fecha_Nacimiento, con un formato dd/mm/aaa.
Fecha_Prestamo	Almacena la fecha en la que se realiza el préstamo cada personal.	Conjunto de cadenas válidas tipo Date para el atributo Fecha_Prestamo, con un formato dd/mm/aaa.
fijo	Almacena el número de teléfono fijo de cada personal.	Conjunto de cadenas válidas para el atributo fijo compuesta por caracteres enteros del 0-9 con un rango de 15 caracteres.
ID_B	Almacena el identificador único de cada baja.	Valor numérico entero positivo en el rango de 1-9999 para el atributo ID_B.
ID_Per	Almacena el identificador único de cada personal.	Valor numérico entero positivo en el rango de 1-9999 para el atributo ID_Per.
ID_Pres	Almacena el identificador único de cada préstamo.	Valor numérico entero positivo en el rango de 1-9999 para el atributo ID_Pres.
ID_Pue	Almacena el identificar único de cada puesto.	Valor numérico entero positivo en el rango de 1-99 para el atributo ID_Pue.
ID_Suc	Almacena el identificador único de cada sucursal.	Valor numérico entero positivo en el rango de 1-99 para el atributo ID_Suc.

motivo	Almacena el motivo por el cual se da de baja a cada personal.	Conjunto de cadenas válidas para el atributo estado compuesta de 500 caracteres de la A-Z con espacios, puntos y acentos.
nom_p	Se almacena el nombre de cada personal.	Conjunto de cadenas válidas para el atributo nom_p compuesta de 25 caracteres de la A-Z con espacios, puntos y acentos.
nombre_suc	Almacena el nombre de cada sucursal.	Conjunto de cadenas válidas para el atributo nombre_suc compuesta de 22 caracteres de la A-Z con espacios, puntos y acentos.
nss	Almacena el número de seguro social de cada personal.	Conjunto de cadenas válidas para el atributo nss compuesta de 12 caracteres de la A-Z con espacios, puntos y acentos.
numero	Almacena el número de casa donde vive el personal.	Conjunto de cadenas válidas para el atributo numero compuesta de 6 caracteres de la A-Z y de 0-9.
rfc	Almacena el rfc de cada personal.	Conjunto de cadenas válidas para el atributo rfc compuesta de 13 caracteres de la A-Z con espacios, puntos y acentos.
tipo_s	Almacena el número de control de cada personal.	Conjunto de cadenas válidas para el atributo tipo_s compuesta de 3 caracteres de la A-Z con espacios, puntos y acentos.
Vencimiento_Contrato	Almacena la fecha de vencimiento de contrato de cada personal.	Conjunto de cadenas válidas tipo Date para el atributo Vencimiento_Contrato, con un formato dd/mm/aaa.
Vencimiento_Licencia	Almacena la fecha de vencimiento de licencia de conducir de cada personal.	Conjunto de cadenas válidas Date para el atributo Vencimiento_Licencia, con un formato dd/mm/aaa.

Tabla 1. Diccionario de datos.

Documentacion Del Código Del Sistema

Modulo Gestionar Personal



Se deberán importar las siguientes librerías ya que son necesarias, las más importantes son la 2da, 3era, 4ta, y 5ta, ya que estas establecen la conexión con la base, y son utilizadas para consultas entre otras cosas.

```
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.Calendar;
import com.mxrck.autocompleter.AutoCompleterCallback;
import com.mxrck.autocompleter.TextAutoCompleter;
import com.toedter.calendar.JTextFieldDateEditor;
import java.awt.Color;
import java.awt.HeadlessException;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Period;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.Date;
import javax.swing.table.TableColumn;
import javax.swing.table.TableColumnModel;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.swing.JTextField;
```

Declaraciones Globales

De forma global se declaran las siguientes variables:

```
DateFormat df=DateFormat.getDateInstance();
static String TPS="O-",P,S,ColCombobox;
int vts=0; //Esta es la variable la cual obtiene el valor del combo box de tipo de sangre
Connection Con;
Statement St;
ResultSet Rs;
ResultSet resultset;
Statement statement;
TextAutoCompleter autocompleta;  //autocompleter del código postal
TextAutoCompleter autoComCampoBusqueda1;
TextAutoCompleter autoComCampoBusqueda2;
String ConsultaNombre="";
MuestraMotivo m=new MuestraMotivo();
boolean punto=false;
Aviso_Contrato avc=new Aviso_Contrato();
Aviso_Licencia2 avl=new Aviso_Licencia2();
```

Método Personal

El método public personal es el método constructor del módulo personal, en él se inicializan los componentes necesarios para el correcto funcionamiento de los botones, calendarios y los autocompleter de los distintos campos que lo requieren.
También se definen las propiedades necesarias para que los autocompleter o calendarios realicen cierta tarea al hacer una selección en ellos de las opciones que ofrecen al usuario.

```
public Personal() {
    initComponents();//inicia el ciclo de vida de los componentes
    setLocationRelativeTo(null);//se define que el frame debe aparecer en el centro de la pantalla
    //se cambia el color del frame a blanco
    getContentPane().setBackground(new java.awt.Color(255,255,255));
    setResizable(false);//se define el jframe como no reajustable
    this.setTitle("Personal - Usuario: "+Globales.User);//se le pone un encabezado al frame
    llenarpuesto();//se llena el JcomboBox de puestos con los puestos existentes.
    llenarsucursal();//se llena el JcomboBox de sucursales con las sucursales existentes.
    nom.requestFocus();//se pone el cursor en el campo del nombre
    autoComCampoBusqueda1=new TextAutoCompleter(busqueda1);//definición del autocompleter del campo
busqueda1

    //se define que al elegir un personal en el campo de busqueda2
    //se llamara al método consulta enviándole el mismo nombre o parámetro
    autoComCampoBusqueda2 = new TextAutoCompleter(busqueda2, new AutoCompleterCallback() {
        @Override
        public void callback(Object selectedItem) {
            consulta((String)selectedItem);
        }
       });

    //se define que al elegir un código postal en el campo de cp
    //se llamara al método Ayuda_de_cp enviándole el mismo código
    //postal como parámetro
    autocompleta = new TextAutoCompleter(cp, new AutoCompleterCallback() {
        @Override
        public void callback(Object selectedItem) {
            Ayuda_de_cp((String)selectedItem);
```

```
        }
       });

//Se agrega una propiedad al jDateChooser de la fecha de nacimiento
      //para que al presionar el día en el calendario se calcule la edad y no tenga que
      //ingresarla el usuario
      FechaNa.getDateEditor().addPropertyChangeListener(new PropertyChangeListener() {
       @Override
       public void propertyChange(PropertyChangeEvent e) {
          //si no esta vacio el campo del calendario
          if(FechaNa.getDate() == null){
              cp.requestFocus();
            }else{
             //como no esta vacio se procede a calcular la edad
             int años=CalculaEdad();//método que retorna los años de edad
             edad.setText(""+años);//se ponen los años en el campo de la edad
             cp.requestFocus();//se redirige el cursor al campo del código postal
            }
       }
      });

      //Se agrega una propiedad al jDateChooser de la fecha del contrato
      //para que al elegir una fecha se active el campo para la fecha de vencimiento de contrato
      //y a su vez defina la fecha permitida para el vencimiento de contrato
      FechaContra.getDateEditor().addPropertyChangeListener(new PropertyChangeListener() {
       @Override
       public void propertyChange(PropertyChangeEvent e) {
          if(FechaContra.getDate() == null){
            //si el campo del calendario es vacío
          }else{
            //si el campo del calendario no es vacío
            FechaVenContra.setEnabled(true);//se activa la fecha de vencimiento de contrato

            //se define el formato correcto de la fecha y se obtiene la fecha del contrato
            //y se guarda como tipo date en la variable: contrato_date
            DateTimeFormatter fmt = DateTimeFormatter.ofPattern("yyyy-M-d");
            String contra=GetFormatoFecha("FechaContra");
            LocalDate contrato = LocalDate.parse(contra, fmt);
            Date contrato_date = Date.from(contrato.atStartOfDay(ZoneId.systemDefault()).toInstant());

            //se define la fecha válida para el vencimiento de contrato a partir de la ya ingresada fecha de contrato
            Calendar cal=Calendar.getInstance();
            cal.setTime(contrato_date);
            cal.add(Calendar.DAY_OF_YEAR,1);
            FechaVenContra.setMinSelectableDate(cal.getTime());
          }
       }
      });

      //Se agrega una propiedad al jDateChooser de la fecha del vencimiento de contrato
      //para solo se pueda elegir una fecha superior a la fecha de contrato
      FechaVenContra.getDateEditor().addPropertyChangeListener(new PropertyChangeListener() {
       @Override
         public void propertyChange(PropertyChangeEvent e) {
          if(FechaVenContra.getDate() != null||FechaVenContra.getCalendar()!=null){
            DateTimeFormatter fmt = DateTimeFormatter.ofPattern("yyyy-M-d");//se establece el formato para la fecha
             String contra=GetFormatoFecha("FechaContra");//se obtiene en la variable contra la fecha de contrato que ya se
ingreso
             LocalDate contrato = LocalDate.parse(contra, fmt);//fecha del contrato en formato Date
             String vencontrato=GetFormatoFecha("FechaVenContra");//se obtiene en la variable vencontrato la fecha del
vencimiento de contrato que se acaba de ingresar
```

```
            LocalDate fecha_ven_contrato = LocalDate.parse(vencontrato,fmt);//fecha del vencimiento del contrato en formato Date
            int correcto=fecha_ven_contrato.compareTo(contrato);//se almacena -1 si el vencimiento de contrato es menor al contrato,se almacena cero si ambas fechas son iguales

            if(correcto<1){
               JOptionPane.showMessageDialog(null,"La fecha de vencimiento de contrato debe ser mayor a la del contrato");
               ((JTextField)FechaVenContra.getDateEditor().getUiComponent()).setText("");
            }
          }

          }
      });

      //método que valida que la fecha de nacimiento no sea en el futuro, y que el empleado tenga más de 15 años y menos de 80
       ValidaNacimiento();

       //se define a estos calendarios que el rango para elegir fecha
       //no se permite si es un dia anterior al actual.
       FechaIngre.setMinSelectableDate(new Date());
       FechaContra.setMinSelectableDate(new Date());
       FechaVenContra.setMinSelectableDate(new Date());

       //lo siguiente es para bloquear los campos de texto de las fechas
       //se crean un objetos del tipo DateEditor y se ponen en false para no poder editar la fecha seleccionada desde el teclado
       JTextFieldDateEditor editorFechaNa,editorFechaContra,editorFechaVenContra,editorFechaVenLM,editorFechaIngre;//se crea el objeto tipo DateEditor

       //se declaran los objetos de tipo editor para los calendarios
       editorFechaNa=(JTextFieldDateEditor) FechaNa.getDateEditor();
       editorFechaContra=(JTextFieldDateEditor) FechaContra.getDateEditor();
       editorFechaVenContra=(JTextFieldDateEditor) FechaVenContra.getDateEditor();
       editorFechaVenLM=(JTextFieldDateEditor) FechaVenLM.getDateEditor();
       editorFechaIngre=(JTextFieldDateEditor) FechaIngre.getDateEditor();

       //se desactivan las ediciones a los objetos.
       editorFechaNa.setEditable(false);
       editorFechaContra.setEditable(false);
       editorFechaVenContra.setEditable(false);
       editorFechaVenLM.setEditable(false);
       editorFechaIngre.setEditable(false);
   }
```

Método nomKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto del nombre para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
   private void nomKeyTyped(java.awt.event.KeyEvent evt) {
      //primero se guarda en la cadena nombre lo que hay en el campo
      //de texto antes del nuevo carácter ingresado
      String nombre=nom.getText();
        //se obtiene solo el carácter ingresado
        char k=evt.getKeyChar();

       //solo si el carácter ingresado es una letra, un punto o un espacio se
       //dejara ingresar y se analizara el orden correcto para la validación
       if(Character.isLetter(k)||k=='.'||k==' '){
```

```
        //si el campo de texto no está vacío
        if(nombre.length()>0){
          //si el último carácter en el campo de texto es un punto
          //y el carácter ingresado es diferente de un espacio
          //entonces no se deja ingresar este carácter
         if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
          getToolkit().beep();
          evt.consume();
         }
         //si el último carácter en el campo de texto es un espacio
         //y el carácter ingresado es un espacio o un punto
         ////entonces no se deja ingresar este carácter
         else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
          getToolkit().beep();
          evt.consume();
         }
         //no se deja ingresar un carácter si en el campo de texto ya hay 25 caracteres
          else if(nombre.length()==25){
            getToolkit().beep();
            evt.consume();
          }
        }
        //si el campo de texto si está vacío no se pueden
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
            getToolkit().beep();
            evt.consume();
          }
        }
       }
       //si el carácter es algo diferente de una letra, punto o espacio
       //no se deja ingresar el carácter
       else{
           evt.consume();
       }
       //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
           getToolkit().beep();
           evt.consume();
        }
}
```

Método apKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto del apellido paterno para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void apKeyTyped(java.awt.event.KeyEvent evt) {
      //primero se guarda en la cadena: nombre lo que hay en el campo
      //de texto antes del nuevo carácter ingresado
       String nombre=ap.getText();
       //se obtiene el nuevo carácter ingresado
        char k=evt.getKeyChar();
       //si el caracter ingresado es letra, punto o espacio este es analizado
       if(Character.isLetter(k)||k=='.'||k==' '){
        if(nombre.length()>0){//si el campo de texto no está vacío
          //si el último carácter en el campo de texto es un punto
           //y el carácter ingresado es diferente de un espacio
```

```
        //entonces no se deja ingresar este carácter
       if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
        getToolkit().beep();
        evt.consume();
       }
       //si el último carácter en el campo de texto es un espacio
       //y el carácter ingresado es un espacio o un punto
       ////entonces no se deja ingresar este carácter
       else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
        getToolkit().beep();
        evt.consume();
       }
       //no se deja ingresar un carácter si en el campo de texto ya hay 25 caracteres
        else if(nombre.length()==25){
        getToolkit().beep();
          evt.consume();
        }
      }
      //si el campo de texto si esta vacío no se pueden
      //ingresar espacios o puntos
      else if(nombre.isEmpty()){
        if(k==' '||k=='.'){
          getToolkit().beep();
          evt.consume();
        }
      }
     }else{
         //getToolkit().beep();
         evt.consume();
     }
     //si el campo de texto ya contiene un punto ya no deja ingresar otro
     if(nombre.contains(".")&&k=='.'){
         getToolkit().beep();
         evt.consume();
      }
}
```

Método amKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto del apellido materno para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void amKeyTyped(java.awt.event.KeyEvent evt) {
    //primero se guarda en la cadena: nombre lo que hay en el campo
   //de texto antes del nuevo carácter ingresado
    String nombre=am.getText();
    //se obtiene el nuevo carácter ingresado
     char k=evt.getKeyChar();
    //si el carácter ingresado es letra, punto o espacio este es analizado
    if(Character.isLetter(k)||k=='.'||k==' '){
     if(nombre.length()>0){//si el campo de texto no está vacío
      //si el ultimo carácter en el campo de texto es un punto
       //y el carácter ingresado es diferente de un espacio
       //entonces no se deja ingresar este carácter
      if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
       getToolkit().beep();
       evt.consume();
      }
      //si el ultimo carácter en el campo de texto es un espacio
      //y el carácter ingresado es un espacio o un punto
```

```
        ////entonces no se deja ingresar este carácter
        else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
         getToolkit().beep();
         evt.consume();
        }
        //no se deja ingresar un carácter si en el campo de texto ya hay 25 caracteres
         else if(nombre.length()==25){
         getToolkit().beep();
           evt.consume();
         }
        }
        //si el campo de texto si esta vacío no se pueden
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
            getToolkit().beep();
            evt.consume();
          }
        }
       }else{
           //getToolkit().beep();
           evt.consume();
       }
       //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
           getToolkit().beep();
           evt.consume();
        }
    }
```

Método calleKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto de la calle para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void calleKeyTyped(java.awt.event.KeyEvent evt) {
       //primero se guarda en la cadena nombre lo que hay en el campo
       //de texto antes del nuevo carácter ingresado
       String nombre=calle.getText();
       //se obtiene solo el carácter ingresado
       char k=evt.getKeyChar();
       //solo si el carácter ingresado es una letra, un digito, un punto o un espacio se
        //dejara ingresar y se analizara el orden correcto para la validación
        if(Character.isLetter(k)||k=='.'||k==' '||Character.isDigit(k)){
          //si el campo de texto no está vacío
          if(nombre.length()>0){
            //si el último carácter en el campo de texto es un punto
            //y el carácter ingresado es diferente de un espacio
            //entonces no se deja ingresar este carácter
            if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
            getToolkit().beep();
            evt.consume();
           }
            //si el último carácter en el campo de texto es un espacio
           //y el carácter ingresado es un espacio o un punto
           ////entonces no se deja ingresar este carácter
           else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
            getToolkit().beep();
            evt.consume();
```

```
        }
        //no se deja ingresar un carácter si en el campo de texto ya hay 50 caracteres
         else if(nombre.length()==50){
           getToolkit().beep();
           evt.consume();
         }
        }
         //si el campo de texto si esta vacío no se pueden
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
           getToolkit().beep();
           evt.consume();
         }
        }
       }
       //si el carácter es algo diferente de una letra, punto o espacio
       //no se deja ingresar el carácter
       else{
           //getToolkit().beep();
           evt.consume();
       }
       //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
           getToolkit().beep();
           evt.consume();
        }
    }
```

Método numeroKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto del número para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
    private void numeroKeyTyped(java.awt.event.KeyEvent evt) {
      //se obtiene el carácter ingresado
       char k=evt.getKeyChar();
      //se obtienen los caracteres que ya están el campo de texto
       String NUMERO=numero.getText();
      //solo se analizaran dígitos y letras
      if(Character.isDigit(k)||Character.isLetter(k)){
       //no se aceptan espacios
        if(k==' '){
         getToolkit().beep();
         evt.consume();
       }
       ////no se acepta un cero como primer carácter
       else if(NUMERO.isEmpty()&&(k=='0'||Character.isLetter(k))){
           getToolkit().beep();
           evt.consume();
       }
       else if(NUMERO.length()>0){
          //no se aceptan mas de 6 números
          if(NUMERO.length()==6){
           getToolkit().beep();
           evt.consume();
          }
          //solo se acepta una letra después de un número
          else if(Character.isLetter(NUMERO.charAt(NUMERO.length()-1))){
           getToolkit().beep();
```

```
            evt.consume();
          }
        }
      }else{
            evt.consume();
      }
    }
```

Método apKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo de texto número de seguro social para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void nssKeyTyped(java.awt.event.KeyEvent evt) {
        //se obtiene el carácter ingresado y los que ya están
        //el campo de texto
        char k=evt.getKeyChar();
        String NSS=nss.getText();
        //solo se aceptan dígitos
        if(Character.isDigit(k)){
            //si el campo de texto no está vacío
            //solo se permiten 11 caracteres
            if(NSS.length()>0){
             if(NSS.length()==11){
               getToolkit().beep();
               evt.consume();
             }
            }
        }
        //no deja entrar caracteres que no sean dígitos
        else{
          evt.consume();
        }

    }
```

Método rfcKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo RFC para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void rfcKeyTyped(java.awt.event.KeyEvent evt) {
        //se obtiene el carácter ingresado
        //y los caracteres que ya están ingresados
        char k=evt.getKeyChar();
        String RFC=rfc.getText();
        //si el caracter ingresado es letra o número se acepta
       if(Character.isDigit(k)||Character.isLetter(k)){
        //no se aceptan espacios
        if(k==' '){
          getToolkit().beep();
          evt.consume();
        }
```

```
        //no se puede meter un número como primer carácter
        else if(RFC.isEmpty()&&(Character.isDigit(k))){
            getToolkit().beep();
            evt.consume();
        }
        //se checa si el campo no está vacío
        else if(RFC.length()>0){
          //acepta 13 caracteres máximo
          if(RFC.length()==13){
           getToolkit().beep();
           evt.consume();
          }
          //no se aceptan dígitos en los primeros 4 caracteres
          else if(RFC.length()<=3&&Character.isDigit(k)){
           getToolkit().beep();
           evt.consume();
          }
          //no se aceptan letras entre los caracteres 5 y 10
          else if(RFC.length()>=4&&RFC.length()<=9&&Character.isLetter(k)){
           getToolkit().beep();
           evt.consume();
          }
        }
       }else{
            evt.consume();
       }
    }
```

Método curpKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo CURP para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void curpKeyTyped(java.awt.event.KeyEvent evt) {
      //se obtienen el carácter ingresado y los que ya están
      //en el campo de texto
      char k=evt.getKeyChar();
      String CURP=curp.getText();
     //solo se analizaran digitos y letras
     if(Character.isDigit(k)||Character.isLetter(k)){
      //no se aceptan espacios
      if(k==' '){
        getToolkit().beep();
        evt.consume();
      }
      //si el campo esta vacío no acepta dígitos
      else if(CURP.isEmpty()&&(Character.isDigit(k))){
          getToolkit().beep();
          evt.consume();
      }
      //si el campo no está vacío
      else if(CURP.length()>0){
        //solo se aceptan 18 caracteres máximo
        if(CURP.length()==18){
         getToolkit().beep();
         evt.consume();
        }
        //no se aceptan dígitos en los primeros 4 caracteres
        else if(CURP.length()<=3&&Character.isDigit(k)){
         getToolkit().beep();
```

```
            evt.consume();
           }
           //no se aceptan letras entre los caracteres 5 y 10
           else if(CURP.length()>=4&&CURP.length()<=9&&Character.isLetter(k)){
            getToolkit().beep();
            evt.consume();
           }
           //no se aceptan dígitos entre los caracteres 12 y 16
           else if(CURP.length()>=10&&CURP.length()<=15&&Character.isDigit(k)){
            getToolkit().beep();
            evt.consume();
           }
           //no se aceptan letras en los últimos 2 caracteres
           else if(CURP.length()>=16&&CURP.length()<=17&&Character.isLetter(k)){
            getToolkit().beep();
            evt.consume();
           }
         }
        }
       //si no es digito o letra no se permite el carácter
       else{
           evt.consume();
       }
    }
```

Método celularKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo celular para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void celularKeyTyped(java.awt.event.KeyEvent evt) {
      //se obtienen el carácter ingresado
      //y los que están en el campo de texto
      char k=evt.getKeyChar();
      String NUMERO=celular.getText();
      //solo se aceptan números
     if(Character.isDigit(k)){
      //no acepta espacios
         if(k==' '){
        getToolkit().beep();
        evt.consume();
       }
       else if(NUMERO.length()>0){
          //solo se aceptan 13 números
           if(NUMERO.length()==13){
           getToolkit().beep();
           evt.consume();
          }
       }
      }
     //no se acepta nada que no sea numero
     else{
         evt.consume();
     }
  }
```

Método fijoKeyTyped

El siguiente método se manda a llamar automáticamente cada vez que se ingresa un carácter en el campo teléfono fijo para validar que solo se puedan ingresar los caracteres validos de acuerdo a la documentación.

```
private void fijoKeyTyped(java.awt.event.KeyEvent evt) {
    //se obtienen el carácter ingresado
    //y los que están en el campo de texto
    char k=evt.getKeyChar();
    String NUMERO=fijo.getText();
     //solo se aceptan números
   if(Character.isDigit(k)){
    if(k==' '){
      getToolkit().beep();
      evt.consume();
    }
     //no acepta espacios
    else if(NUMERO.length()>0){
        //solo se aceptan 10 números
       if(NUMERO.length()==10){
        getToolkit().beep();
        evt.consume();
       }
    }
   }
   //no se acepta nada que no sea numero
   else{
       evt.consume();
   }
}
```

Método ValidaNacimiento

Este método valida que el calendario de la fecha de nacimiento solo permita elegir fechas de 15 a 80 años atrás de la fecha actual, este se manda a llamar en el método constructor.

```
/método que valida el calendario de fecha de nacimiento
   public void ValidaNacimiento(){
     Calendar cal = Calendar.getInstance();
     cal.add(Calendar.YEAR, -15);//15 años antes de la fecha actual
     Date min = cal.getTime();
     cal.add(Calendar.YEAR,-65);//65+15=80 años antes de la fecha actual
     Date max = cal.getTime();
     FechaNa.setSelectableDateRange(max,min);//ahora solo se puede seleccionar entre 80 y 15 años antes
   }
```

Método ValidaContrato

Este método valida que el calendario de la fecha de contrato solo permita elegir fechas a partir de la fecha actual y no de fechas pasadas, este se manda a llamar en el método constructor.

```
   //método que valida el calendario de la fecha de contrato
   public void ValidaContrato(){
     LocalDate ahora = LocalDate.now();//la fecha del día actual
     //el rango mínimo de selección es el día actual
     Date min = Date.from(ahora.atStartOfDay(ZoneId.systemDefault()).toInstant());
     Date max=min;
     FechaContra.setSelectableDateRange(max,min);//ahora solo se puede elegir a partir del día actual
```

```
}
```

Método CalculaEdad

Este método calcula la edad en años tomando como base la fecha que este seleccionada en el calendario de fecha de nacimiento, se usa para poner la edad automáticamente cuando se está agregando un nuevo personal o bien, cuando se consulta un personal se compara si la edad calculada coincide con la registrada, esto con el motivo de actualizar la edad de un personal que ya tiene tiempo laborando en la empresa.

```
//método que calcula los años de edad de acuerdo a la fecha que este
//registrada o se seleccione
public int CalculaEdad(){
        //se obtiene la fecha y se guarda en la cadena nacimiento
        String nacimiento=GetFormatoFecha("FechaNa");
        DateTimeFormatter fmt = DateTimeFormatter.ofPattern("yyyy-M-d");//se hace un objeto fmt para el formato deseado a manejar
        LocalDate fechaNac = LocalDate.parse(nacimiento,fmt);//fechaNac tiene la fecha de nacimiento
        LocalDate ahora = LocalDate.now();//la fecha de hoy se guarda en la variable ahora
        Period periodo = Period.between(fechaNac,ahora);//se calcula el tiempo que ha pasado entre la fecha de nacimiento y hoy
        return periodo.getYears();//se retornan los años calculados
}
```

Método FormWindowOpened

Este método se ejecuta inmediatamente al abrir la ventana del módulo personal, aquí hay varios métodos que se ejecutan a modo de inicializar la tabla de personal, agregar los registros de los códigos postales a la lista del autocompleter del campo de texto para el código postal, ajusta la búsqueda especializada con puestos y nombre y activa las alertas de contrato y licencia.

```
///este método se ejecuta al abrir el frame
private void formWindowOpened(java.awt.event.WindowEvent evt) {
  ConstruyeTabla();//especifica tamaños y propiedades de la tabla para el personal
  cargardatosTabla("CALL CDTP;");//llena la tabla del personal
  autocompletaDatosCP();//carga los códigos postales a la lista del campo de códigos postales
  //ajusta la busqueda especializada por defecto
  autoComCampoBusqueda1("Puesto");
  autoComCampoBusqueda2("Nombre");
  //alertas de contrato y licencia
  DFV("select ID_Per,nom_p,ap,am,Vencimiento_Contrato,datediff(Vencimiento_Contrato,curdate()) as dias from personal where datediff(Vencimiento_Contrato,curdate())<8 and condicion='activo'");
  DFL("select ID_Per,nom_p,ap,am,Vencimiento_Licencia,datediff(Vencimiento_Licencia,curdate()) as diasl from personal where datediff(Vencimiento_Licencia,curdate())<8 and condicion='activo'");
}
```

Método cargardatos

Este método se manda a llamar cuando re realiza una consulta a la base de datos y se requiere cargar los resultados a los campos de texto, calendarios y listas desplegables.

```
public void cargardatos(String TP)
{
```

```
    try{
       Rs.beforeFirst();//se reinicia el resulset
       colonia.removeAllItems();//se limpia la lista de colonias
       while(Rs.next())
       {
        //se cargan los datos encontrados en los campos
        //de texto correspondientes y se ajustan los calendarios
        id.setText(Rs.getString(1));
        nom.setText(Rs.getString(4));
        ap.setText(Rs.getString(5));
        am.setText(Rs.getString(6));
        Date date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(7));FechaNa.setDate(date);
        edad.setText(Rs.getString(8));
        tiposangre.setSelectedItem(TP);//STRING 9
        calle.setText(Rs.getString(10));
        numero.setText(Rs.getString(11));
        colonia.addItem(Rs.getString(12));
        cp.setText(Rs.getString(13));
        ciudad.setText(Rs.getString(14));
        estado.setText(Rs.getString(15));
        nss.setText(Rs.getString(16));
        rfc.setText(Rs.getString(17));
        curp.setText(Rs.getString(18));
        celular.setText(Rs.getString(19));
        fijo.setText(Rs.getString(20));
        //ajuste de calendarios
        date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(21));FechaIngre.setDate(date);
        date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(22));FechaContra.setDate(date);
        date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(23));FechaVenContra.setDate(date);
        date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(24));FechaVenLM.setDate(date);
       }
      }catch(Exception e){
       JOptionPane.showMessageDialog(null,e);
      }
    }
```

Método llenaTablaBajas

Este método se manda a llamar cuando realiza una consulta a la base de datos y se requiere cargar los registros de las bajas en la tabla de historial de bajas.

```
//Para llenar la tabla de bajas con los datos del personal
    public void llenaTablaBajas(){
    try{
       //se define el modelo de la tabla y se hace conexión a la base de datos
        DefaultTableModel tablab=new DefaultTableModel();
        Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
        St=Con.createStatement();
        //se consulta si el personal tiene bajas registradas
        //Rs=St.executeQuery("select * from Bajas where ID_Per ="+Integer.parseInt(id.getText())+";");
        Rs=St.executeQuery("CALL CTDB('"+Integer.parseInt(id.getText())+"');");
        tablab.setColumnIdentifiers(new Object[]{"ID_B","Motivo","Fecha De Baja"});
        //se rellena la tabla con las bajas
        while(Rs.next())
         {
          tablab.addRow(new Object[]{Rs.getString(1),Rs.getString(3),Rs.getString(4)});
         }
        jTable2.setModel(tablab);
        //se cierra la conexión con la base de datos
        St.close();
        Rs.close();
        Con.close();
```

```
    }catch(Exception e){}
  }
```

Método DFV

Este método se manda a llamar cada vez que se abre la ventana del módulo personal, checa si hay algún trabajador que tenga una fecha de vencimiento de contrato con una semana de cercanía para mostrar una alerta en una ventana que muestre los datos básicos del trabajador y el tiempo que le quedan a su contrato.

```
//alerta contrato
   public void DFV(String consulta)
   {
      int DiasF=0;
    try {
         //se hace conexión a la base de datos
         Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
         St=Con.createStatement();
         Rs=St.executeQuery(consulta);
         //si cierra la ventana si esta ya está abierta
         if(Rs.last()){
           if (avc!=null) {//si existe una venta, la cierra.
             avc.dispose();
           }
            avc.setVisible(true);
         }
         //se cierra la conexión de la base de datos
         St.close();
         Rs.close();
         Con.close();
      }catch (Exception e) {
         JOptionPane.showMessageDialog(null,"Error: "+e);
      }
   }
```

Método DFL

Este método se manda a llamar cada vez que se abre la ventana del módulo personal, checa si hay algún trabajador que tenga una fecha de vencimiento de licencia con una semana de cercanía para mostrar una alerta en una ventana que muestre los datos básicos del trabajador y el tiempo que le queda a su licencia.

```
   //alerta licencia
   public void DFL(String consulta)
   {
      int DiasF=0;
    try {
         //se hace conexión a la base de datos
         Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
         St=Con.createStatement();
         Rs=St.executeQuery(consulta);
         //si cierra la ventana si esta ya esta abierta
         if(Rs.last()){
           if (avl!=null) {//si existe una venta, la cierra.
             avl.dispose();
```

```
            }
             avl.setVisible(true);
            //new Aviso_Licencia2().setVisible(true);
           }
           //se cierra la conexión a la base de datos
           St.close();
           Rs.close();
           Con.close();
       }catch (Exception e) {
          JOptionPane.showMessageDialog(null,"Error: "+e);
       }
    }
```

Método jTable2MouseClicked

Este método se manda a llamar cada vez se efectúa un click izquierdo en alguno de los registros que muestra la tabla de historia de bajas y se abre una pequeña ventana que muestra a manera de texto del motivo por el cual el personal se dio de baja.

```
private void jTable2MouseClicked(java.awt.event.MouseEvent evt) {
        int filaseleccionada;
        try{
              //se obtiene el número de renglón seleccionado
              filaseleccionada= jTable2.getSelectedRow();
              DefaultTableModel modelotabla=(DefaultTableModel) jTable2.getModel();
              //se obtiene el motivo de baja de la tabla
              String motivotb=(String)modelotabla.getValueAt(filaseleccionada, 1);

              if (m!= null) {//si existe una venta, la cierra.
                m.dispose();
              }
               m.setVisible(true);
               //se manda a llamar una ventana que muestra el motivo de baja
               MuestraMotivo.motivoF.setText(motivotb);

        }catch (HeadlessException ex){
           JOptionPane.showMessageDialog(null, "Error: "+ex+"\nInténtelo nuevamente", " .::Error En la Operacion::."
,JOptionPane.ERROR_MESSAGE);
        }
    }
```

Método busqueda2MouseClicked

Este método se manda a llamar cada vez se efectúa un click izquierdo en el campo de texto "busqueda2" el cual es el principal campo para la búsqueda, este actualiza su mismo autocompleter para la búsqueda predictiva y rellena la tabla con los parámetros de búsqueda que estén seleccionados.

```
private void busqueda2MouseClicked(java.awt.event.MouseEvent evt) {
        SeleccionCombo2();//esto para que actualice el autocompleter del texfield busqueda2
        //y a continuación llenamos la tabla dependiendo de lo que se haya seleccionado en el combobox1 y si es activo o
inactivo
        busqueda_especial_tabla();
    }
```

Método activosMouseClicked

Este método se manda a llamar cada vez se efectúa un click izquierdo en la casilla "activos", este rellena la tabla con los parámetros de búsqueda que estén seleccionados.

```
private void activosMouseClicked(java.awt.event.MouseEvent evt) {
      //llenamos la tabla dependiendo de lo que se haya seleccionado en el combobox1 y si es activo o inactivo
       busqueda_especial_tabla();
   }
```

Método inactivosMouseClicked

Este método se manda a llamar cada vez se efectúa un click izquierdo en la casilla "inactivos", este rellena la tabla con los parámetros de búsqueda que estén seleccionados.

```
   private void inactivosMouseClicked(java.awt.event.MouseEvent evt) {
      //llenamos la tabla dependiendo de lo que se haya seleccionado en el combobox1 y si es activo o inactivo
      busqueda_especial_tabla();
   }
```

Método GetFormatoFecha

Este método se manda a llamar cada vez que se ocupe obtener un formato de fecha correcto de los calendarios que existen en el módulo Personal. Es decir, cada vez que se quiere obtener la fecha de algún calendario se manda a llamar este método, se le envía como parámetro el nombre del calendario y se recibe la fecha del mismo en su formato correcto.

```
//este método retorna el formato correcto de fecha de algún calendario
   public String GetFormatoFecha(String nombDateChooser){
     //nombDateChooser es un parámetro que sirve para saber que calendario
     //se debe tomar para sacar el formato
      //se declaran las variables que se van a usar en el método
      String formato="";
      String dia,mes,año;
      //si se requiere la fecha del formato de nacimiento
      if(nombDateChooser.equals("FechaNa")){
          dia = Integer.toString(FechaNa.getCalendar().get(Calendar.DAY_OF_MONTH));
          mes = Integer.toString(FechaNa.getCalendar().get(Calendar.MONTH) + 1);
          año = Integer.toString(FechaNa.getCalendar().get(Calendar.YEAR));
          formato = (año + "-" + mes+ "-" + dia);
      }
      //si se requiere la fecha de ingreso
      else if(nombDateChooser.equals("FechaIngre")){
          dia = Integer.toString(FechaIngre.getCalendar().get(Calendar.DAY_OF_MONTH));
          mes = Integer.toString(FechaIngre.getCalendar().get(Calendar.MONTH) + 1);
          año = Integer.toString(FechaIngre.getCalendar().get(Calendar.YEAR));
          formato = (año + "-" + mes+ "-" + dia);
      }
      //si se requiere la fecha de contrato
      else if(nombDateChooser.equals("FechaContra")){
          dia = Integer.toString(FechaContra.getCalendar().get(Calendar.DAY_OF_MONTH));
          mes = Integer.toString(FechaContra.getCalendar().get(Calendar.MONTH) + 1);
          año = Integer.toString(FechaContra.getCalendar().get(Calendar.YEAR));
          formato = (año + "-" + mes+ "-" + dia);
```

```
        }
        //si se requiere la fecha de vencimiento de contrato
        else if(nombDateChooser.equals("FechaVenContra")){
            dia = Integer.toString(FechaVenContra.getCalendar().get(Calendar.DAY_OF_MONTH));
            mes = Integer.toString(FechaVenContra.getCalendar().get(Calendar.MONTH) + 1);
            año = Integer.toString(FechaVenContra.getCalendar().get(Calendar.YEAR));
            formato = (año + "-" + mes+ "-" + dia);
        }
        //si se requiere la fecha de vencimiento de licencia
        else if(nombDateChooser.equals("FechaVenLM")){
            dia = Integer.toString(FechaVenLM.getCalendar().get(Calendar.DAY_OF_MONTH));
            mes = Integer.toString(FechaVenLM.getCalendar().get(Calendar.MONTH) + 1);
            año = Integer.toString(FechaVenLM.getCalendar().get(Calendar.YEAR));
            formato = (año + "-" + mes+ "-" + dia);
        }
        //se retorna el correcto formato de  del calendario requerido
        return formato;
    }
```

Método Construye Tabla

El siguiente método sirve para crear una tabla de forma manual.

```
    public void ConstruyeTabla(){
        jTable1.setAutoResizeMode(jTable1.AUTO_RESIZE_OFF);//No definir su tamaño automáticamente.
        TableColumn columnas = jTable1.getColumnModel().getColumn(0); //Se define el número de columnas.
        columnas.setPreferredWidth(1);
    }
```

Método Cargar Datos Tabla

El siguiente método funciona para que una vez ingresado a este módulo la tabla se llene con los diferentes datos que tienen el personal.

```
public void cargardatosTabla(String consulta)
    {
//Se crean las variables a utilizar
        String PUESTO="",SUCURSAL="";
        int ID_PUE=0,ID_SUC=0;
        ResultSet resultset;
        Statement statement;
        DefaultTableModel tabla=new DefaultTableModel(); //Se crea un objeto de tipo tabla.
      try{
         Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);//Se realiza la conexión
         St=Con.createStatement();
         statement=Con.createStatement();
         Rs=St.executeQuery(consulta);
         tabla.setColumnIdentifiers(new Object[]{"ID_Per","Nombre","Apellido Paterno","Apellido Materno","Edad","Tipo
de Sangre","Ciudad","Estado","Puesto","Sucursal","Fecha
Ingreso","Contratacion","Ven_Contra","Ven_Lic","Condicion"}); //Se crea la tabla con sus identificadores.
         while(Rs.next())
         {
          ID_PUE=Integer.parseInt(Rs.getString("ID_Pue"));
          ID_SUC=Integer.parseInt(Rs.getString("ID_Suc"));

          resultset=statement.executeQuery("select nom_p from puesto where ID_Pue = "+ID_PUE+";");
          while(resultset.next())PUESTO=resultset.getString("nom_p");

          resultset=statement.executeQuery("select nom_suc from sucursal where ID_Suc = "+ID_SUC+";");
          while(resultset.next())SUCURSAL=resultset.getString("nom_suc");
```

```
            resultset.close();
            tabla.addRow(new
Object[]{Rs.getString(1),Rs.getString(4),Rs.getString(5),Rs.getString(6),Rs.getString(8),Rs.getString(9),Rs.getString(14),
Rs.getString(15),PUESTO,SUCURSAL,Rs.getString(21),Rs.getString(22),Rs.getString(23),Rs.getString(24),Rs.getString
(25)});     //Se mandan los datos a sus respectivos lugares en la tabla.
          }
          jTable1.setModel(tabla);

          statement.close();
          St.close();
          Rs.close();
          Con.close(); //Se cierra la conexión
         }catch(Exception e){}
    }
```

Método Llenar puesto

El siguiente método es utilizado para llenar el combobox de puestos que se tienen en la base de datos.

```
    public void llenarpuesto(){
      puesto.removeAllItems();
      String consulta="select*from puesto";    //Selecciona todo de puesto
        try {
            Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Realiza conexión
            St=Con.createStatement();
            Rs=St.executeQuery(consulta);
            while(Rs.next()){
              puesto.addItem(Rs.getString("nom_p"));  //Añade el nombre del puesto a combobox
            }
            St.close();
            Rs.close();
            Con.close(); //Cierra la conexión
        }catch (Exception e) {
           JOptionPane.showMessageDialog(null,"Error: "+e);  //Si hay un error se muestra un mensaje de error y cual error
        }
    }
```

Método llenarsucursal

El siguiente método es utilizado para llenar el combobox de sucursales que se tienen en la base de datos.

```
    public void llenarsucursal(){ //Este método llena el combobox de sucursal
      sucursal.removeAllItems();   //Se remueven todos los items que se tenían
      String consulta="select*from sucursal";  //Se selecciona todo de sucursal
        try {
            Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza la conexión con la
base de datos.
            St=Con.createStatement();
            Rs=St.executeQuery(consulta);
            while(Rs.next()){
              sucursal.addItem(Rs.getString("nom_suc"));  //Se añaden los nombres de las sucursales al combobox
            }
            St.close();
            Rs.close();
            Con.close(); //Se cierra la conexión con la base de datos
        }catch (Exception e) {
```

```
        JOptionPane.showMessageDialog(null,"Error: "+e); //Si se produce un error se muestra un mensaje de error y cual error fue.
    }
  }
```

Método autocompletaDatosCp

El siguiente método es utilizado para añadir al método autocompletar el código postal todos los códigos postales que se tienen en la base de datos.

```
  public void autocompletaDatosCP(){
    //autocompleta= new TextAutoCompleter(cp);
    try {
        Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Realiza conexión a la base de datos.
        St=Con.createStatement();
        Rs=St.executeQuery("SELECT CodigoPostal from codigospostales;");  //Selecciona  solo el código postal de codigos postales
        while(Rs.next()){
            autocompleta.addItem(Rs.getString("CodigoPostal"));     //Añade al combobox el código postal mientras se tengan resultados
        }
        St.close();
        Rs.close();
        Con.close();  //Se cierra conexión a la base de datos.
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,e);  //Si se produce un error se muestra un mensaje de error y se muestra cual error fue.
    }
  }
```

Método Busqueda_especial_tabla

El siguiente método realiza búsquedas especializadas dependiendo de lo que se selecciones en los combobox, y de las casillas marcadas en activos e inactivos.

```
  public void busqueda_especial_tabla(){
    boolean act=false,inact=false;
    if(activos.isSelected()){act=true;}
    if(inactivos.isSelected()){inact=true;}

    if(busqueda1.getText().isEmpty()){//si esta vacio el campo de búsqueda 1 entonces que ignore si es activo o inactivo
      if(act==true&&inact==true||act==false&&inact==false)//Si ambos están activos o inactivos
        //cargardatosTabla("select * from Personal;");
      cargardatosTabla("CALL CDTP;");//Busca todo de personal
      else if(act==true&&inact==false)//Si activo esta marcado pero inactivos esta desmarcada
        //cargardatosTabla("select * from Personal where condición='Activo';");
      cargardatosTabla("CALL `TODO_ACTIVO`();"); //Busca todo de personal que sea activo.
      else if(act==false&&inact==true)   //Si activo esta desmarcada y inactivo está marcada
        //cargardatosTabla("select * from Personal where condición='Inactivo';");
      cardgardatosTabla("CALL `TODO_INACTIVO`();"); //Selecciona todo del personal que sea inactivo.
     }
    else{
```

```
        if(act==true&&inact==true||act==false&&inact==false){//ignora si es activo o inactivo
            if(ComboBusqueda1.getSelectedItem()=="Puesto")//Si la selección es puesto
//cargardatosTabla("select * from Personal,puesto where Puesto.nom_p = "'+busqueda1.getText()+'" and
personal.ID_Pue=puesto.ID_Pue;");
            cargardatosTabla("CALL TODO_PUE("'+busqueda1.getText()+'");"); //Selecciona todo de personal donde el
nombre del puesto sea igual al del campo de búsqueda y donde el ID de puesto del personal sea el ID del puesto.
            else if(ComboBusqueda1.getSelectedItem()=="Sucursal")
//cargardatosTabla("select * from Personal,Sucursal where sucursal.nom_suc = "'+busqueda1.getText()+'" and
personal.ID_Suc=sucursal.ID_Suc;");
            cargardatosTabla("CALL TODO_SUC("'+busqueda1.getText()+'");//Selecciona todo del personal donde el
nombre del puesto sea igual al del campo de busqueda y donde el ID de sucursal del personal sea el ID del sucursal.
            else if(ComboBusqueda1.getSelectedItem()=="Estado")
//cargardatosTabla("select * from Personal where estado = "'+busqueda1.getText()+'";");
            cargardatosTabla("CALL TODO_ESTADO("'+busqueda1.getText()+'");//Selecciona todo del personal donde
el nombre del estado sea igual al del campo de búsqueda y donde el ID de estado del personal sea el ID del estado.
        }
        else if(act==true&&inact==false){
            if(ComboBusqueda1.getSelectedItem()=="Puesto")
//cargardatosTabla("select * from Personal,puesto where Puesto.nom_p = "'+busqueda1.getText()+'" and
personal.condicion='Activo' and personal.ID_Pue=puesto.ID_Pue;");
            cargardatosTabla("CALL TODO_PUE_ACT("'+busqueda1.getText()+'");//Selecciona todo de personal
donde el nombre del puesto sea igual al del campo de búsqueda y donde el ID de puesto del personal sea el ID del puesto
y donde condicion sea activo
            else if(ComboBusqueda1.getSelectedItem()=="Sucursal")
//cargardatosTabla("select * from Personal,Sucursal where sucursal.nom_suc = "'+busqueda1.getText()+'" and
personal.condicion='Activo' and personal.ID_Suc=sucursal.ID_Suc;");
            cargardatosTabla("CALL TODO_SUC_ACT("'+busqueda1.getText()+'");//Selecciona todo del personal
donde el nombre del puesto sea igual al del campo de búsqueda y donde el ID de sucursal del personal sea el ID del
sucursal y donde condicion sea activo
            else if(ComboBusqueda1.getSelectedItem()=="Estado")
//cargardatosTabla("select * from Personal where estado = "'+busqueda1.getText()+'" and condición='Activo';");
            cargardatosTabla("CALL TODO_ESTADO_ACT("'+busqueda1.getText()+'");//Selecciona todo del personal
donde el nombre del estado sea igual al del campo de búsqueda y donde el ID de estado del personal sea el ID del estado y
donde condición sea activo
        }
        else if(act==false&&inact==true){
            if(ComboBusqueda1.getSelectedItem()=="Puesto")
//cargardatosTabla("select * from Personal,puesto where Puesto.nom_p = "'+busqueda1.getText()+'" and
personal.condicion='Inactivo' and personal.ID_Pue=puesto.ID_Pue;");
            cargardatosTabla("CALL TODO_PUE_INACT("'+busqueda1.getText()+'");//Selecciona todo de personal
donde el nombre del puesto sea igual al del campo de búsqueda y donde el ID de puesto del personal sea el ID del puesto
y donde condición sea inactivo
            else if(ComboBusqueda1.getSelectedItem()=="Sucursal")
//cargardatosTabla("select * from Personal,Sucursal where sucursal.nom_suc = "'+busqueda1.getText()+'" and personal.
Condición='Inactivo' and personal.ID_Suc=sucursal.ID_Suc;");
            cargardatosTabla("CALL TODO_SUC_INACT("'+busqueda1.getText()+'");//Selecciona todo del personal
donde el nombre del puesto sea igual al del campo de busqueda y donde el ID de sucursal del personal sea el ID del
sucursal y donde condicion sea inactivo
            else if(ComboBusqueda1.getSelectedItem()=="Estado")
//cargardatosTabla("select * from Personal where estado = "'+busqueda1.getText()+'" and condición='Inactivo';");
            cargardatosTabla("CALL TODO_ESTADO_INACT("'+busqueda1.getText()+'");//Selecciona todo del
personal donde el nombre del estado sea igual al del campo de búsqueda y donde el ID de estado del personal sea el ID del
estado y donde condición sea inactivo
        }//else if
    }//else
}
```

Método nomKeyPressed

El siguiente método es utilizado para validar que el campo nombre no este vacío, contenga espacios al inicio o al final del mismo, todo esto al momento de presionar estas teclas en el teclado.

```
private void nomKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        int tamaño=nom.getText().length();//Se iguala la variable tamaño al tamaño del campo nom
        if(nom.getText().isEmpty()){//Si el campo nom esta vacío
            ap.requestFocus();//Cambia el cursor al campo apellido paterno
            labelnombre.setForeground(Color.black);
        }
        else if(String.valueOf(nom.getText().charAt(0)).equals(" ")){// sino, se convierte el campo nombre a string y se compara con espacio
            JOptionPane.showMessageDialog(null,"Su primera letra no debe ser un espacio en blanco");//Si es asi, muestra el mensaje
            labelnombre.setForeground(Color.red);//Coloca el color de letra rojo para indicar error
        }
        else if(String.valueOf(nom.getText().charAt(nom.getText().length() - 1)).equals(" ")){//Si no convierte a String el campo nombre, y si su ultimo carácter menos 1 es espacio
            JOptionPane.showMessageDialog(null,"Su ultima letra no debe ser un espacio en blanco");//Muestra el mensaje
            labelnombre.setForeground(Color.red);//Coloca el color de letra rojo para indicar error
        }
        else {//Si ninguna condición se cumple
            ap.requestFocus();//Mueve el cursor al campo apellido paterno
            labelnombre.setForeground(Color.black);
        }
    }
}
```

Método apKeyPressed

El siguiente método es utilizado para validar que el campo apellido paterno no este vacío, contenga espacios al inicio o al final del mismo, todo esto al momento de presionar estas teclas en el teclado.

```
private void apKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER ){
      int tamaño=ap.getText().length();//Se iguala la variable tamaño al tamaño del campo ap
        if(ap.getText().isEmpty()){//Si el campo ap esta vacío
            am.requestFocus();//Mueve el cursor al campo apellido materno
            labelap.setForeground(Color.black);
        }
        else if(String.valueOf(ap.getText().charAt(0)).equals(" ")){ //Sino convierte a strinng el valor del campo ap en su primer carácter y este es igual a espacio
            JOptionPane.showMessageDialog(null,"Su primera letra no debe ser un espacio en blanco");//imprime este mensaje
            labelap.setForeground(Color.red);//Cambia el color de letra a rojo para indicar error
        }
        else if(String.valueOf(ap.getText().charAt(ap.getText().length() - 1)).equals(" ")){ // Sino convierte a string el valor del ampo ap en su último carácter menos uno y si este es igual a espacio
            JOptionPane.showMessageDialog(null,"Su ultima letra no debe ser un espacio en blanco"); // Muestra el mensaje
            labelap.setForeground(Color.red); //Cambia el color de letra a rojo para indicar error.
        }
        else {
```

```
            am.requestFocus(); // Si no se cumple ninguna condición cambia el cursor al campo am
            labelap.setForeground(Color.black);
        }
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){
      nom.requestFocus(); //Mueve el cursor al campo nombre
    }
}
```

Método amKeyPressed

El siguiente método es utilizado para validar que el campo apellido materno no este vacío, contenga espacios al inicio o al final del mismo, todo esto al momento de presionar estas teclas en el teclado.

```
private void amKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){ //Si la tecla presionada fue enter
      int tamaño=am.getText().length();//Se iguala la variable tamaño al tamaño del campo am
        if(am.getText().isEmpty()){//Si apellido materno está vacío
            edad.requestFocus();//Mueve el cursor al campo edad
            labelam.setForeground(Color.black);//Cambia el color de la letra a negra
        }
        else if(String.valueOf(am.getText().charAt(0)).equals(" ")){//Sino convierte a strinng el valor del campo am en su primer carácter y este es igual a espacio
            JOptionPane.showMessageDialog(null,"Su primera letra no debe ser un espacio en blanco");//Si esta vacío muestra el mensaje
            labelam.setForeground(Color.red);//Cambia a color rojo la letra para indicar error
        }
        else if(String.valueOf(am.getText().charAt(am.getText().length() - 1)).equals(" ")){// Sino convierte a string el valor del ampo am en su ultimo caracter menos uno y si este es igual a espacio
            JOptionPane.showMessageDialog(null,"Su ultima letra no debe ser un espacio en blanco");//Muestra el mensaje
            labelam.setForeground(Color.red);//Cambia el color de la letra a rojo para indicar error
        }
        else {
            edad.requestFocus();//Cambia el cursor al campo de edad
            labelam.setForeground(Color.black);//Cambia el color de la letra a color negro
        }
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){//Si la tecla presionada fue arriba
    ap.requestFocus();//Cambia el cursor al campo de ap
    }
}
```

Método edadKeyPressed

El siguiente método es utilizado para mover el cursor a otro campo dependiendo de cuál sea la tecla presionada.

```
private void edadKeyPressed(java.awt.event.KeyEvent evt) {
  if(evt.getKeyCode()==KeyEvent.VK_ENTER){     //Si la tecla presionada fue enter
      cp.requestFocus();// Cambia el cursor al campo de código postal(cp)
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){//Si la tecla presionada fue arriba
      am.requestFocus();//Cambia el cursor a apellido materno (am)
    }
}
```

Método calleKeyPressed

El siguiente método es utilizado para mover el cursor a otro campo dependiendo de cuál sea la tecla presionada.

```
private void calleKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){ //Si la tecla presionada fue enter
      numero.requestFocus(); //Cambia el cursor al campo numero
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){ //Si la tecla presionada fue arriba
      ciudad.requestFocus(); //Cambia el cursor al campo de ciudad
    }
}
```

Método numeroKeyPressed

El siguiente método es utilizado para mover el cursor a otro campo dependiendo de cuál sea la tecla presionada.

```
private void numeroKeyPressed(java.awt.event.KeyEvent evt) {
  if(evt.getKeyCode()==KeyEvent.VK_ENTER){  //Si la tecla presionada es enter
    nss.requestFocus();  //cambia el cursor al campo de número de seguro social (nss)
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){ // Si la tecla presionada fue arriba
      calle.requestFocus(); // Cambia el cursor al campo de calle.
    }
}
```

Método cpKeyPressed

El siguiente método es utilizado para mover el cursor a otro campo dependiendo de cuál sea la tecla presionada, e invoca un método dependiendo de que el campo este vacío o no.

```
private void cpKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){       // Si la tecla presionada fue enter
                    estado.requestFocus();        //Cambia el cursor al campo de estado
      if(cp.getText().isEmpty()){} // Si el campo cp esta vacío no hagas nada
      else Ayuda_de_cp(cp.getText());       //Pero si no está vacío, invoca el metodo Ayuda_de_cp con lo que tenga el
campo de cp
    }
}
```

Método Ayuda_de_cp

El siguiente método es utilizado para llenar el combobox con todas las colonias, municipios, y estado que se tienen con el código postal ingresado.

```
public void Ayuda_de_cp(String codigo){
    colonia.removeAllItems(); //se elimina todo lo que tenga el jcombobox a modo de reinicio
    try {
       Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza conexión a la base de
datos
       St=Con.createStatement();
Rs=St.executeQuery("select Estado,Municipio from codigospostales where CodigoPostal = "+codigo+";"); //Se realiza un
result set con una búsqueda, la cual selecciona el estado, el municipio de código postal, donde el código postal sea igual al
código postal enn el campo de cp
       while(Rs.next()){
            estado.setText(Rs.getString("Estado")); //Obtiene el Estado
```

```
                ciudad.setText(Rs.getString("Municipio"));     //Obtiene el municipio
            }
            String col=""; //Se crea la variable col
            Rs=St.executeQuery("select Colonia from codigospostales where CodigoPostal = "+codigo+";"); //Se realiza un result set con una búsqueda, la cual selecciona la colonia de código postal donde el código postal sea igual al ingresado
            while(Rs.next()){
                col=(Rs.getString("Colonia")); // Se iguala la variable col al resultado de la Colonia
            }
            char[]caracteres; // Se crea un arreglo de caracteres de tipo char
            caracteres=col.toCharArray(); // Se iguala el arreglo de caracteres a la conversión de la variable col a arreglo de caracteres
            col="";          //Se iguala la variable col a vacío
            for (int x = 0; x < caracteres.length; x++) { // Mientras que la variable x sea menor al tamaño del arreglo caracteres
              if(caracteres[x]==';'){      // Si caracteres en la posición, del valor de x es igual a punto y coma
               colonia.addItem(col); //Añade al combobox la colonia
               col="";  //Iguala la variable col a vacío
              }else{                          //Sino
               col+=Character.toString(caracteres[x]);     //Iguala la variable col a la conversión del arreglo en la posición del arreglo en el valor de x
                                    if(x==caracteres.length-1)colonia.addItem(col);  //Si el valor de x es igual al tamaño del arreglo menos uno, añade la colonia al combobox de colonia
              }
            }
            St.close();
            Rs.close();
            Con.close(); //Se cierra la conexión con la base de datos
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null,e);  //Si se tiene una excepción muestra cual fue.
        }
    }
```

Método ciudadKeyPressed

El siguiente método es utilizado para mover el cursor a ciertos campos dependiendo de cuál sea la tecla presionada.

```
    private void ciudadKeyPressed(java.awt.event.KeyEvent evt) {
        if(evt.getKeyCode()==KeyEvent.VK_ENTER){ //Si la tecla presionada fue enter
          calle.requestFocus(); //Cambia el cursor al campo de calle
        }
        else if(evt.getKeyCode()==KeyEvent.VK_UP){  // Si la tecla presionada fue arriba
          estado.requestFocus();  // Cambia el cursor al campo de estado
        }
    }
```

Método estadoKeyPressed

El siguiente método es utilizado para validar el campo de estado, para evitar que este vacío, contenga espacios al inicio o al final.

```
    private void estadoKeyPressed(java.awt.event.KeyEvent evt) {
        if(evt.getKeyCode()==KeyEvent.VK_ENTER){ // Sil la tecla presionada fue enter
            int tamaño=estado.getText().length();    //Iguala la variable tamaño al tamaño del campo estado
            if(estado.getText().isEmpty()){    //Si el campo estado están vacío
                ciudad.requestFocus();  //Cambia el cursor al campo ciudad
                labelestado.setForeground(Color.black); //Cambia el color de la letra a negro
            }
            else if(String.valueOf(estado.getText().charAt(0)).equals(" ")){  //Convierte el valor de estado a String y si su primer caracter es espacio
                JOptionPane.showMessageDialog(null,"Su primera letra no debe ser un espacio en blanco");  //Muestra el mensaje
```

```
            labelestado.setForeground(Color.red);  //Cambia el color de letra a rojo para indicar error
        }
        else if(String.valueOf(estado.getText().charAt(estado.getText().length() - 1)).equals(" ")){  //Convierte el valor de estado a String y si su ultimo caracter menos uno es espacio
            JOptionPane.showMessageDialog(null,"Su ultima letra no debe ser un espacio en blanco");  //Muestra el siguiente mensaje
            labelestado.setForeground(Color.red);  //Camba el color de la letra a rojo para indicar error
        }
        else {  //Sino
            ciudad.requestFocus();  //Cambia el cursor al campo ciudad
            labelestado.setForeground(Color.black);  //Cambia el color de la letra a negro
        }
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){  //Si la tecla presionada fue arriba
     edad.requestFocus();  //Cambia el cursor al campo edad
    }
  }
```

Método nssKeyPressed

El siguiente método es utilizado para mover el cursor dependiendo de la tecla presionada en el campo de número de seguro social.

```
 private void nssKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){   //Si la tecla presionada fue enter
     rfc.requestFocus(); //Cambia el cursor al campo rfc
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){   //Si la tecla presionada fue arriba
     numero.requestFocus(); //Cambia el cursor al campo número
    }
  }
```

Método rfcKeyPressed

El siguiente método es utilizado para mover el cursor dependiendo de la tecla presionada en el campo de RFC.

```
   private void rfcKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){  //Si la tecla presionada fue enter
     curp.requestFocus();   //Cambia el cursor al campo de curp
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){  //Si la tecla presionada fue arriba
     nss.requestFocus();  //cambia el cursor al campo de número de seguro social.
    }
  }
```

Método curpKeyPressed

El siguiente método es utilizado para mover el cursor dependiendo de la tecla presionada en el campo de CURP.

```
  private void curpKeyPressed(java.awt.event.KeyEvent evt) {
   if(evt.getKeyCode()==KeyEvent.VK_ENTER){   // Si la tecla presionada fue enter
     celular.requestFocus();   //Cambia el cursor al campo celular
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){  //Si la tecla presionada fue arriba
     rfc.requestFocus();  //Cambia el cursor al campo rfc
    }
  }
```

Método celularKeyPressed

El siguiente método es utilizado para mover el cursor dependiendo de la tecla presionada en el campo de celular.

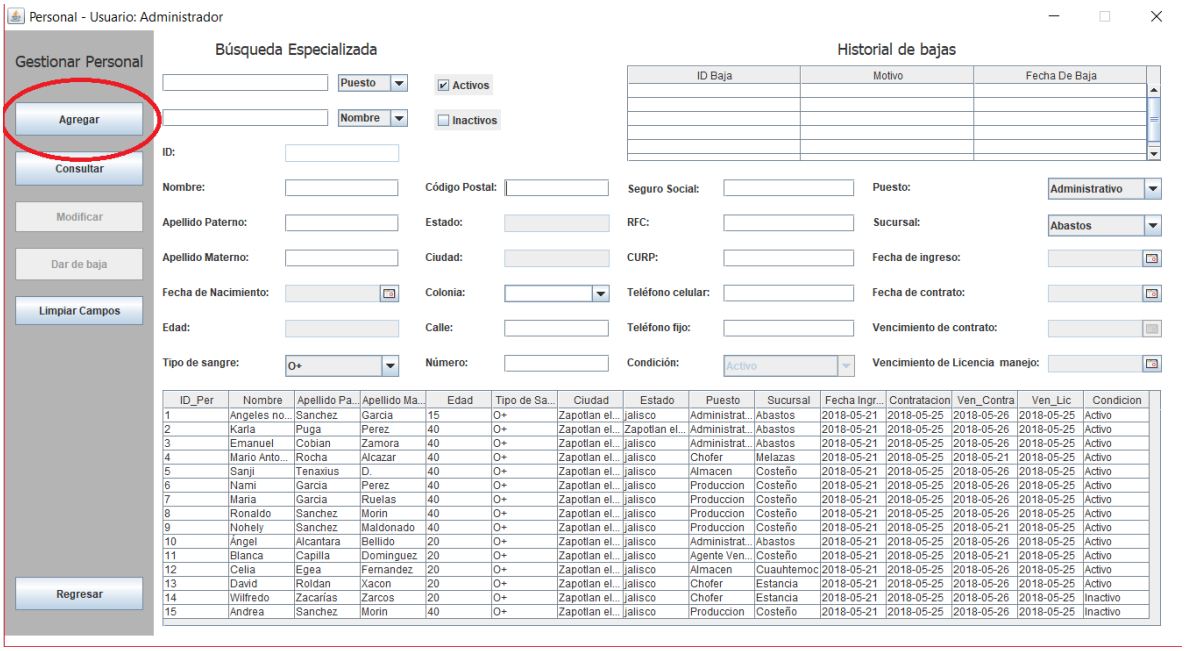
```
private void celularKeyPressed(java.awt.event.KeyEvent evt) {
   if(evt.getKeyCode()==KeyEvent.VK_ENTER){   //Si la tecla presionada fue enter
     fijo.requestFocus();  //Cambia el cursor al campo fijo
    }
    else if(evt.getKeyCode()==KeyEvent.VK_UP){   //Si la tecla presionada fue arriba
     curp.requestFocus();  //Cambia el cursor al campo curp
    }
}
```

Método fijoKeyPressed

El siguiente método es utilizado para mover el cursor dependiendo de la tecla presionada en el campo de fijo.

```
private void fijoKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_UP){   // Si la tecla presionada fue arriba
      celular.requestFocus();  //cambia el cursor al campo celular.
    }
}
```

Botón Agregar Personal



El botón agregar personal manda a llamar al método agregarActionPerformed. Lo primero que realiza este método es validar que no haya ningún campo vacío, después valida que no haya espacios de más en los campos de Nombre, Apellido Paterno, Apellido Materno y

Estado, ya que de estos depende la consulta y no debe haber espacios demás. También se valida que no meta números de teléfono o celular inválidos.
Ya que se validaron los campos ahora es necesario validar que no exista un registro con ese mismo nombre para evitar errores de consulta, para lo que realiza la conexión a la base datos, misma que se usara luego para insertar de esto se obtiene el formato correcto de las fechas ingresadas llamando al método "GetFormatoFecha" para poder ingresarlas a la base de datos.
Una vez hecho lo anterior se inserta en la base de datos todos los datos ya validados que ingreso el usuario y se manda llamar al método "cargardatosTabla" para actualizar la tabla con el nuevo registro que se acaba de agregar.
Se avisa que el registro fue agregado correctamente, se manda a llamar al método "limpiacampos" para limpiar todos los campos, se limpia el autocompleter de la colonia, se desactivan los botones de modificar y dar de baja y se cierran las conexiones a la base de datos.

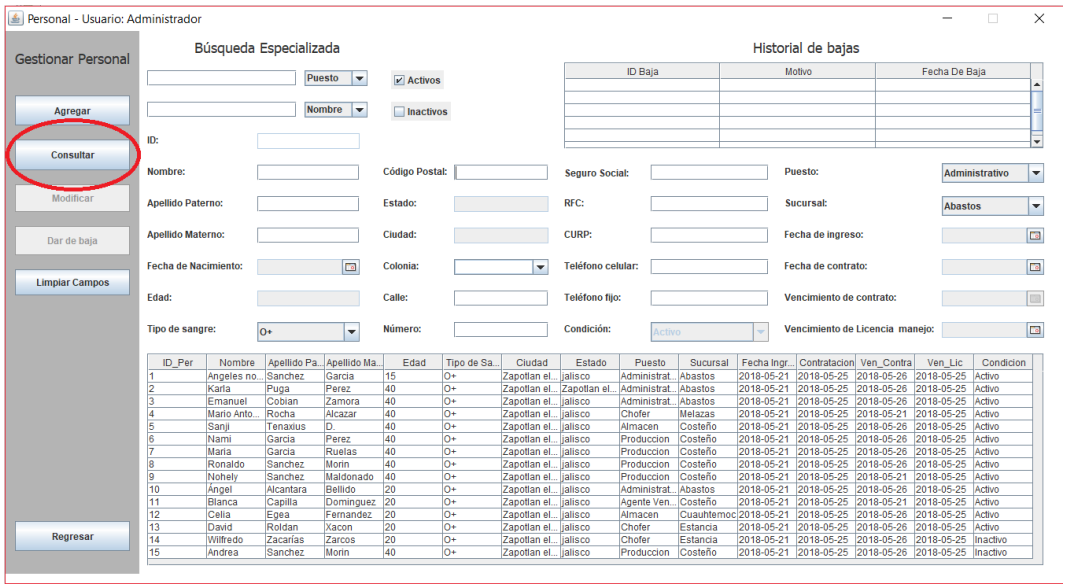
Código:

```
private void agregarActionPerformed(java.awt.event.ActionEvent evt) {
      //Se declaran las variables y banderas  que se van a usar en el método
      int auxedad=0,auxcp=0,idpue=0,idsuc=0;
      String nacimiento,ingreso,contra,ven_contra,ven_lm;
      ColCombobox=(String)colonia.getSelectedItem();

      //Se activan las banderas que nos diran si los campos de texto para id están vacíos o no
      if(edad.getText().isEmpty())auxedad=1;
      else {int EDAD=Integer.parseInt(edad.getText());}
      if(cp.getText().isEmpty())auxcp=1;
      else {int CP=Integer.parseInt(cp.getText());}

     //Se valida que no haya campos vacíos

if(nom.getText().isEmpty()||ap.getText().isEmpty()||am.getText().isEmpty()||FechaNa.getDate()==null||calle.getText().isE
mpty()||numero.getText().isEmpty()||ciudad.getText().isEmpty()||estado.getText().isEmpty()||nss.getText().isEmpty()||rfc.
getText().isEmpty()||curp.getText().isEmpty()||celular.getText().isEmpty()||fijo.getText().isEmpty()||FechaIngre.getDate()
==null||FechaContra.getDate()==null||FechaVenContra.getDate()==null||FechaVenLM.getDate()==null||auxedad==1||au
xcp==1||ColCombobox.isEmpty()){
      JOptionPane.showMessageDialog(null,"Campos vacíos");
     }//if (validar campos vacíos)
     else{
        //validamos que no tenga espacios de más en los campos necesarios para la búsqueda
        //o que los números de teléfonos si tengan su longitud mínima
        boolean correcto=true,minimo=true,minimoF=true;

        if(String.valueOf(nom.getText().charAt(0)).equals("
")||String.valueOf(nom.getText().charAt(nom.getText().length() - 1)).equals(" "))
        {
          labelnombre.setForeground(Color.red);correcto=false;
        }
        if(String.valueOf(ap.getText().charAt(0)).equals(" ")||String.valueOf(ap.getText().charAt(ap.getText().length() -
1)).equals(" "))
        {
          labelap.setForeground(Color.red);correcto=false;
```

```
        }
        if(String.valueOf(am.getText().charAt(0)).equals(" ")||String.valueOf(am.getText().charAt(am.getText().length() -
1)).equals(" "))
        {
          labelam.setForeground(Color.red);correcto=false;
        }
        if(String.valueOf(estado.getText().charAt(0)).equals("
")||String.valueOf(estado.getText().charAt(estado.getText().length() - 1)).equals(" "))
        {
          labelestado.setForeground(Color.red);correcto=false;
        }
        if(celular.getText().length()<10)
        {
          labelcelular.setForeground(Color.red);
          minimo=false;
        }
        if(fijo.getText().length()<7)
        {
          labelfijo.setForeground(Color.red);
          minimoF=false;
        }
        //ahora si se checan las variables booleanas para determinar si hubo espacios de más
        //o si la longitud del número de teléfono o celular es invalido a la especificación
        if(correcto==false||minimo==false||minimoF==false){
          if(correcto==false)JOptionPane.showMessageDialog(null,"Hay espacios en blanco de mas");
          if(minimo==false)JOptionPane.showMessageDialog(null,"La longitud de un numero de celular debe ser minima
de 10 y maxima de 13");
          if(minimoF==false)JOptionPane.showMessageDialog(null,"La longitud de un numero de telefono fijo debe ser
minima de 7 y maxima de 10");
        }
        else{//al entrar aquí quiere decir que almenas las entradas de los datos son correctos
            //Se le pregunta al usuario si está seguro de agregar al registro
            Object [] opciones ={"Aceptar","Cancelar"};
            int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estás seguro de agregar un nuevo
registro?","Mensaje de Confirmación",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");

            if (eleccion == JOptionPane.YES_OPTION)//si la elección fue si
            {
               try{
                  //Se obtiene el nombre completo que se acaba de ingresar
                  String nombre=(nom.getText()+" "+ap.getText()+" "+am.getText());
                  Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
                  St=Con.createStatement();

                  //se checa si este nombre ya existe en la base de datos
                  Rs = St.executeQuery("SELECT  CONCAT(nom_p,' ',ap,' ',am) as Repito FROM personal WHERE
CONCAT(nom_p,' ',ap,' ',am) = '"+nombre+"';");
                  if(Rs.last()){
                     //como si existe se le avisa al usuario y no se agrega
                     JOptionPane.showMessageDialog(null,"Ya existe un registro con ese nombre");
                  }
                  else{
                     //al entrar aquí quiere decir que el nombre no existe y se puede agregar

                     //necesitamos encontrar el id de puesto y sucursal con el nombre elegido
                      Rs=St.executeQuery("CALL SIDP('"+P+"');");
                     while(Rs.next())idpue=Integer.parseInt(Rs.getString("ID_Pue"));
                     Rs=St.executeQuery("CALL SIDS('"+S+"');");
                     while(Rs.next())idsuc=Integer.parseInt(Rs.getString("ID_Suc"));
```

```
                    //obtenemos los formatos correctos de las fechas
                    nacimiento=GetFormatoFecha("FechaNa");
                    ingreso=GetFormatoFecha("FechaIngre");
                    contra=GetFormatoFecha("FechaContra");//de la tabla contratos
                    ven_contra=GetFormatoFecha("FechaVenContra");//de la tabla contratos
                    ven_lm=GetFormatoFecha("FechaVenLM");

                    //ahora si a insertar en la tabla personal
                    String query="insert into
Personal(ID_Pue,ID_Suc,nom_p,ap,am,Fecha_Nacimiento,edad,tipo_s,calle,numero,colonia,cp,ciudad,
estado,nss,rfc,curp,cel,fijo,Fecha_Ingreso,Fecha_Contrato,Vencimiento_Contrato,Vencimiento_Licencia,condicion)
values
("+idpue+","+idsuc+","+nom.getText()+"","+ap.getText()+"","+am.getText()+"","+nacimiento+"","+edad.getText()+"","
+TPS+"","+calle.getText()+"","+numero.getText()+"","+ColCombobox+"","+cp.getText()+"","+ciudad.getText()+"","+es
tado.getText()+"","+nss.getText()+"","+rfc.getText()+"","+curp.getText()+"","+celular.getText()+"","+fijo.getText()+"","
+ingreso+"","+contra+"","+ven_contra+"","+ven_lm+"','Activo');";
                    St.executeUpdate(query);//insercion del personal


                    cargardatosTabla("CALL CDTP");//se actualizan los datos con el nuevo personal ingresado
                    JOptionPane.showMessageDialog(null,"Registro agregado exitosamente");
                    limpiacampos();//se limpian los campos
                    colonia.removeAllItems();//se limpia el autocompleter del campo colonia
                    //se desactivan los botones modificar y dar de baja
                    modificar.setEnabled(false);
                    baja.setEnabled(false);
                   }//else de si existe ya el registro o no

                //se cierra la conexión de la base de datos
                Con.close();
                Rs.close();
                St.close();

                //se actualiza el autocompleter de la búsqueda para consulta
                SeleccionCombo2();
            } catch (Exception e){
               //si hay un error aquí se muestra
               JOptionPane.showMessageDialog(null,e);
            }
        }//if de confirmación
      }//else de espacios en blanco de mas
    }//else de campos vacíos
  }
```

Botón Consultar Personal



El botón consultar personal manda a llamar a su método actionperformed el cual obtiene el parámetro de búsqueda de lo que se requiere consultar para luego mandar a llamar al método "consulta" enviándole este parámetro que se acaba de obtener. El método consulta lo primero que hace es validar que el campo de búsqueda no este vacío, realiza la conexión a la base de datos, checa si el registro existe para continuar con la consulta, de lo contrario se salta toda la consulta, cierra conexiones y finaliza el método.
Una vez que se determinó que el registro si existe se consultan los id del puesto y sucursal de este trabajador para después poder buscar su nombre y seleccionarlo en los jcombobox correspondientes. Después se realiza la consulta de todos los demás datos del trabajador, se manda a llamar al método que llena la tabla del historial de bajas "llenaTablaBajas" y se activa el botón de dar de baja al trabajador solo si su condición esta activa. Finalmente se cierran las correcciones.

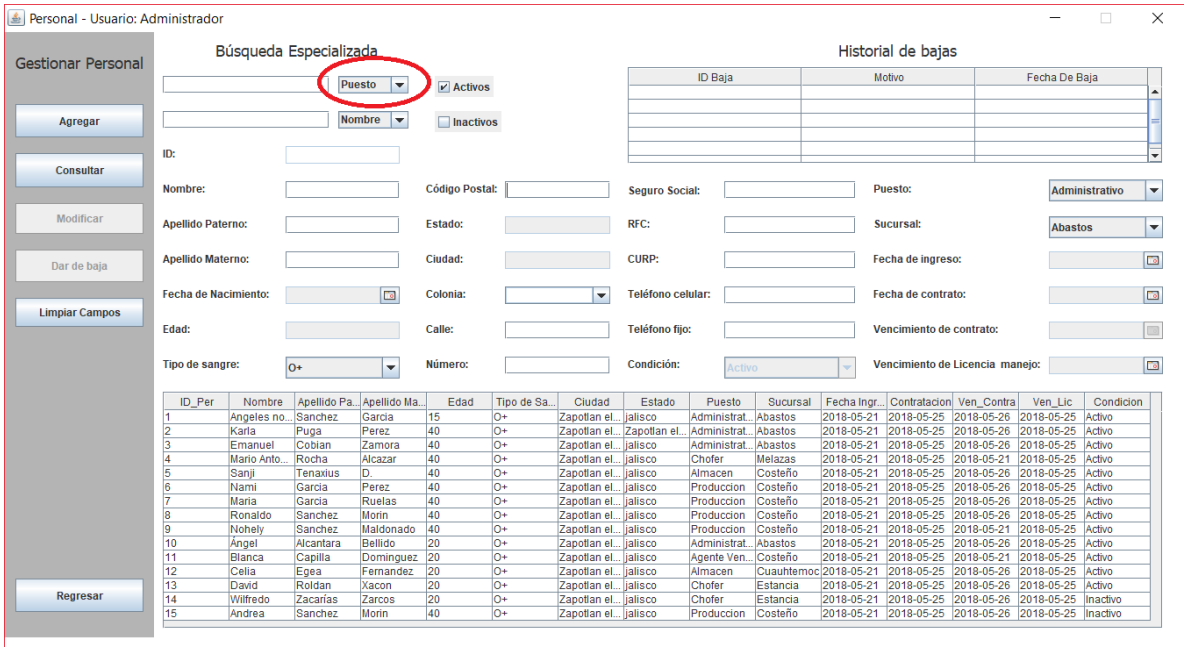
Código

```
private void consultarActionPerformed(java.awt.event.ActionEvent evt) {
      //se obtiene el nombre o id del personal que se quiere consultar
      String nombre=busqueda2.getText();
      //se llama al método que consulta
      consulta(nombre);
   }

   public void consulta(String NOMBRE){
     //primero se verifica que el campo de búsqueda no este vacío
     if(NOMBRE.isEmpty())JOptionPane.showMessageDialog(null,"Campo de búsqueda vacío");
      else{
          boolean existe;
          try{

             //Se realiza conexión a la base de datos
             Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
              St=Con.createStatement();

               //se selecciona la consulta adecuada para ver si el registro existe
```

```
  if(ComboBusqueda2.getSelectedItem()=="Nombre")
  Rs=St.executeQuery("CALL CNC('"+NOMBRE+"');");
  else if(ComboBusqueda2.getSelectedItem()=="Apellido")
  Rs = St.executeQuery("CALL SNCA('"+NOMBRE+"');");
  else if(ComboBusqueda2.getSelectedItem()=="ID"){
  //se quita el espacio que se agrego para consultar sin errores
  NOMBRE=quitaEspacios(NOMBRE);
  Rs = St.executeQuery("CALL SPID('"+NOMBRE+"');");
  }

//a continuación nos damos cuenta si existe o no
if(Rs.last()){existe=true;}
else{
  JOptionPane.showMessageDialog(null,"No existe el registro");
  existe=false;
}

if(existe==true){//en este punto si existe el registro
    //ahora se selecciona la consulta adecuada para buscar en la base de datos
    if(ComboBusqueda2.getSelectedItem()=="Nombre")
    Rs=St.executeQuery("CALL CNCNR('"+NOMBRE+"');");
    else if(ComboBusqueda2.getSelectedItem()=="Apellido")
    Rs=St.executeQuery("CALL          SNCASR('"+NOMBRE+"');");
    else if(ComboBusqueda2.getSelectedItem()=="ID")
    Rs = St.executeQuery("CALL SPIDSR('"+NOMBRE+"');");

    //se preparan las variables necesarias para sacar los id de
    //puestos y sucursales
    String IDPUESTO="",IDSUCURSAL="";
    String PUESTO="",SUCURSAL="",TP="",CONDICION="";

    //ahora sí, se obtienen los id de puestos y sucursales
    while(Rs.next()){
     IDPUESTO=Rs.getString("ID_Pue");
     IDSUCURSAL=Rs.getString("ID_Suc");
     TP=Rs.getString("tipo_s");
     CONDICION=Rs.getString("condicion");
    }
   cargardatos(TP);//se cargan los datos en los campos de texto

    //se revisa si la edad coincide con la fecha de nacimiento
    int edadreal=CalculaEdad();
    int edadregistrada=Integer.parseInt(edad.getText());
    //si la edad no coincide con la fecha de nacimiento entonces se actualiza
    if(edadreal!=edadregistrada){
                                St.executeUpdate("update Personal set edad = "+edadreal+" where ID_Per =
          "+Integer.parseInt(id.getText())+";");
                                 }

    //ahora con los id de puesto y sucursal obtenemos el nombre de estos
    statement=Con.createStatement();
    resultset=statement.executeQuery("select nom_p from puesto where ID_Pue = "+IDPUESTO+";");
    while(resultset.next())PUESTO=resultset.getString("nom_p");
    //se consultan los demás datos del trabajador en la base de datos
    resultset=statement.executeQuery("select nom_suc from sucursal where ID_Suc = "+IDSUCURSAL+";");
    while(resultset.next())SUCURSAL=resultset.getString("nom_suc");
    //se cierran conexiones
    resultset.close();
    statement.close();
    //se seleccionan los puestos y sucursales encontrados en los jcombobox
    puesto.setSelectedItem(PUESTO);
    sucursal.setSelectedItem(SUCURSAL);
```

```
            //se detecta si la condición del trabajador es activa o inactiva
            //para activar o desactivar el botón de dar de baja y mostrarle
            //la condicion del trabajado al empleado
            if(CONDICION.equals("Activo")){
               condicion.setSelectedItem("Activo");
               condicion.setEnabled(false);
               baja.setEnabled(true);
               llenaTablaBajas();
            }
            else if(CONDICION.equals("Inactivo")){
               condicion.setSelectedItem("Inactivo");
               condicion.setEnabled(true);
               baja.setEnabled(false);
               //se llena la tabla del historial de bajas con todas las bajas que se
               //le hayan hecho al trabajador
               llenaTablaBajas();
            }
            //se activa el botón de modificar porque ya se consultó correctamente
            modificar.setEnabled(true);
            busqueda2.setText("");
         }
         //se cierran conexiones
         St.close();
         Rs.close();
         Con.close();
        }catch(Exception e){
          JOptionPane.showMessageDialog(null, e);
        }
      }
  }
```

Búsqueda especializada, ComboBusqueda1, primer parámetro para búsqueda combinada



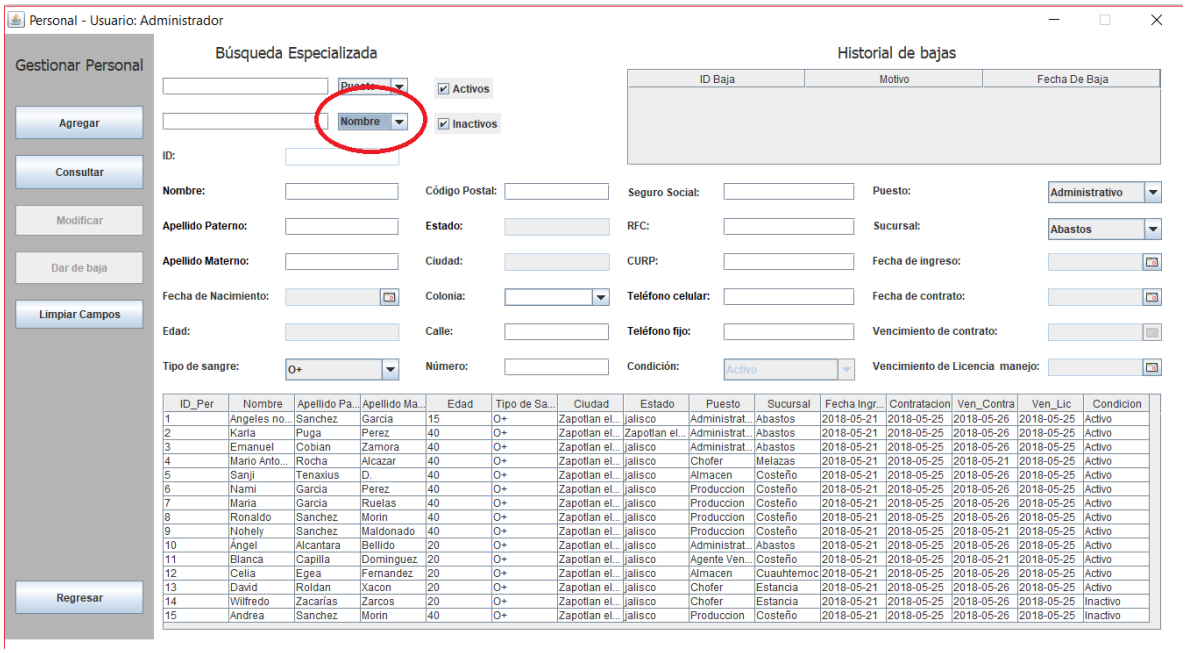
Al presionar el jcombobox "ComboBusqueda1" se manda a llamar a su método actionperformed el cual limpia el cambo de busqueda1 y manda a llamar al método

"SeleccionCombo1" el cual decide que parámetro se debe enviar al método "autoComCampoBusqueda". Este último método es el que rellena la lista del autocompleter del campo busqueda1 ya sea con los puestos, sucursales o estados existentes que estén en uso por algún trabajador.

Código:

```
private void ComboBusqueda1ActionPerformed(java.awt.event.ActionEvent evt) {
      //se limpia el campo de busqueda1 y se manda a llamar al método SeleccionCombo1
      busqueda1.setText("");
      SeleccionCombo1();
   }

   //este método decide que parámetro enviar al método autoComCampoBusqueda1
   //se invoca a este método cuando se quiere actualizar la búsqueda predictiva
   //del campo de busqueda1
   public void SeleccionCombo1(){
      if(ComboBusqueda1.getSelectedItem()=="Puesto")autoComCampoBusqueda1("Puesto");
      else if(ComboBusqueda1.getSelectedItem()=="Sucursal")autoComCampoBusqueda1("Sucursal");
      else if(ComboBusqueda1.getSelectedItem()=="Estado")autoComCampoBusqueda1("Estado");
   }

   //este método rellena al campo de busqueda1 con los registros de los puestos o sucursales o estados registrados
   public void autoComCampoBusqueda1(String parametro){
      //primero se limpia para evitar repeticiones
      autoComCampoBusqueda1.removeAllItems();
     try{
        //se hace conexión a la base de datos.
         Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
         St=Con.createStatement();

         //se decide la consulta que se la hará la base de datos de acuerdo al parámetro recibido
         if(parametro.equals("Puesto")){Rs = St.executeQuery("select  nom_p as Todos from puesto;");}
         else if(parametro.equals("Sucursal")){ Rs = St.executeQuery("select  nom_suc as Todos from sucursal;");}
         else if(parametro.equals("Estado")){Rs = St.executeQuery("select distinct estado as Todos from personal;");}

         //se rellena la lista de la búsqueda predictiva con el resultado de la base de datos
         while(Rs.next())autoComCampoBusqueda1.addItem(Rs.getString("Todos"));

         //se cierran conexiones
         Rs.close();
         St.close();
         Con.close();
       } catch (Exception e){
          //si hay algún error aquí se muestra
               JOptionPane.showMessageDialog(null,e);
       }
   }
```

Búsqueda especializada, ComboBusqueda2, segundo parámetro para búsqueda combinada.



Al presionar el jcombobox "ComboBusqueda2" se manda a llamar a su método actionperformed el cual limpia el cambo de busqueda2 y manda a llamar al método "SeleccionCombo2" el cual decide que parámetro se debe enviar al método "autoComCampoBusqueda2". Este último método es el que rellena la lista del autocompleter del campo busqueda2 con los nombres de los trabajadores, empezando por nombre o por apellido así como también por su ID. Estos métodos analizaran si se solicita o no el primer parámetro de búsqueda que contiene a los puestos, sucursales y estados que están registrados en el personal y también checan las casillas de activos e inactivos para así crear 18 filtros diferentes de búsqueda.

Código:

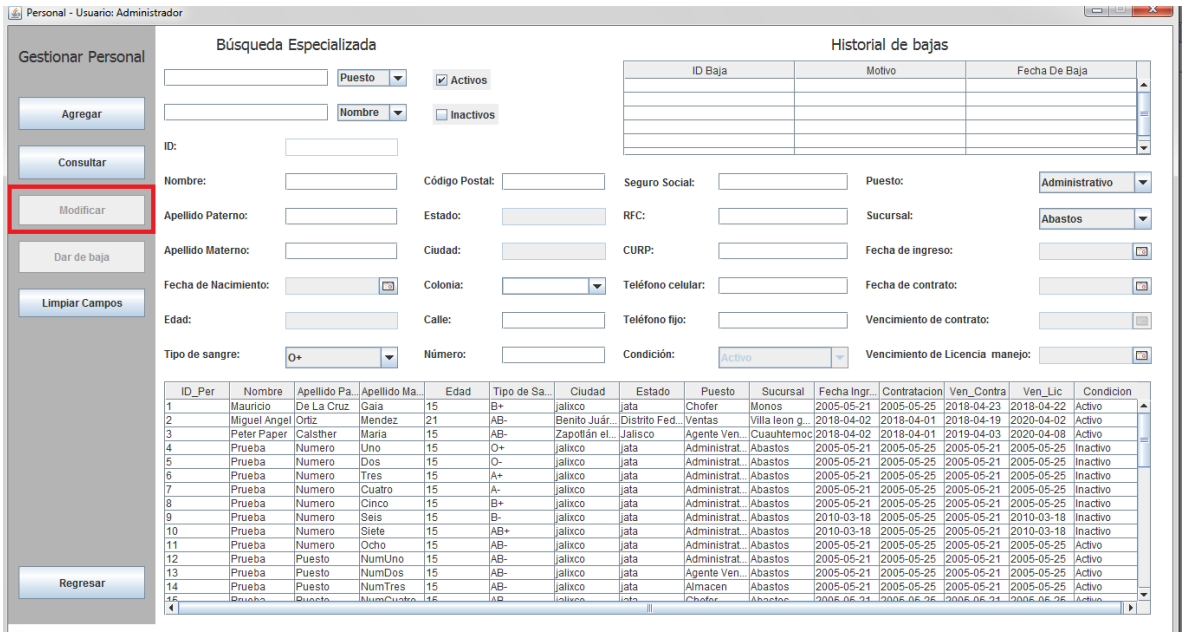
```
private void ComboBusqueda2ActionPerformed(java.awt.event.ActionEvent evt) {
      //se limpia el segundo campo de búsqueda y se manda a llamar al método SeleccionCombo2
      busqueda2.setText("");
      SeleccionCombo2();
   }

   //Este método decide que parámetro enviarle al método autoComCampoBusqueda2
   //de acuerdo a lo que este seleccionado en el jcombobox ComboBusqueda2
   public void SeleccionCombo2(){
      if(ComboBusqueda2.getSelectedItem()=="Nombre")autoComCampoBusqueda2("Nombre");
      else if(ComboBusqueda2.getSelectedItem()=="Apellido")autoComCampoBusqueda2("Apellido");
      else if(ComboBusqueda2.getSelectedItem()=="ID")autoComCampoBusqueda2("ID");
   }

    public void autoComCampoBusqueda2(String parametro){
     //primero se limpia para evitar repeticiones
       autoComCampoBusqueda2.removeAllItems();
```

```
    try{
       //se hace conexión a la base de datos
        Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
        St=Con.createStatement();

        //checa si el campo de busqueda1 esta vacío para ver qué tipo de búsqueda se debe hacer
        if(busqueda1.getText().isEmpty()){//si esta vacío el campo de búsqueda 1

           //checa las casillas de activos e inactivos para ver qué tipo de búsqueda se debe hacer
           boolean act=false,inact=false;
            if(activos.isSelected()){act=true;}
            if(inactivos.isSelected()){inact=true;}

            //ya que se checaron los parámetros de búsqueda que se solicitaron
            //ahora es necesario decidir la búsqueda que se hará con estos
            //parámetros

            //si están seleccionadas ambas casillas, activos e inactivos, o bien, ninguna esta seleccionada
            //después checa si la busqueda1 es por nombre, apellido o Id
            if(act==true&&inact==true||act==false&&inact==false){//ignora si es activo o inactivo para el autocompleter
              if(parametro.equals("Nombre")){Rs = St.executeQuery("CALL ALL_PER_NAME;");}
              else if(parametro.equals("Apellido")){ Rs = St.executeQuery("CALL ALL_PER_AP;");}
              else if(parametro.equals("ID")){Rs = St.executeQuery("CALL ALL_PER_ID;");}
            }
            //si esta seleccionada la casilla activos pero no la de inactivos
            //después checa si la busqueda1 es por nombre, apellido o Id
            else if(act==true&&inact==false){//solo los activos se utilizaran en la búsqueda del autocompleter
              if(parametro.equals("Nombre")){Rs = St.executeQuery("CALL ALL_PER_NAME_ACT;");}
              else if(parametro.equals("Apellido")){ Rs = St.executeQuery("CALL ALL_PER_AP_ACT;");}
              else if(parametro.equals("ID")){Rs = St.executeQuery("CALL ALL_PER_ID_ACT;");}
            }
            //si esta seleccionada la casilla inactivos pero no la de activos
            //después checa si la busqueda1 es por nombre, apellido o Id
            else if(act==false&&inact==true){//solo los activos se utilizaran en la búsqueda del autocompleter
              if(parametro.equals("Nombre")){Rs = St.executeQuery("CALL ALL_PER_NAME_INACT;");}
              else if(parametro.equals("Apellido")){ Rs = St.executeQuery("CALL ALL_PER_AP_INACT");}
              else if(parametro.equals("ID")){Rs = St.executeQuery("CALL ALL_PER_ID_INACT;");}
            }
        }
        //si ya de plano se quiere combinar a todo lo anterior el parámetro de busqueda1
        //el cual puede ser un puesto, sucursal o estado entonces hay que hacer una búsqueda supercombinada
        else{
          BusquedaCombinada();
        } //ahora si se realiza la búsqueda y se rellena el autocompleter
        while(Rs.next()){
           //si se seleccionó la búsqueda por ID es necesario agregar un espacio para evitar errores
           if(ComboBusqueda2.getSelectedItem()=="ID")autoComCampoBusqueda2.addItem(Rs.getString("Todos")+"
");
           else autoComCampoBusqueda2.addItem(Rs.getString("Todos"));
        }
        //se cierran las conexiones a la base de datos
        Rs.close();
        St.close();
        Con.close();
       } catch (Exception e){
             JOptionPane.showMessageDialog(null,e);
       }
   }

   //el siguiente método se invoca cuando se requiere una búsqueda que incluya el parámetro de del cambo busqueda1 y
busqueda2
   //el cual puede ser un puesto, sucursal o estado
```

```
public void BusquedaCombinada(){
  //este método fue hecho para hacer una búsqueda personalizada con los dos combobox de búsqueda
  //se utiliza en autoComCampoBusqueda2
   try {
        //checa las casillas de activos e inactivos para ver que tipo de búsqueda se debe hacer
        boolean act=false,inact=false;
        if(activos.isSelected()){act=true;}
        if(inactivos.isSelected()){inact=true;}

        //Si están seleccionadas ambas casillas, activos e inactivos, o bien, ninguna esta seleccionada
        if(act==true&&inact==true||act==false&&inact==false){
          //búsqueda combinada para puesto
          //Este caso es para cuando esta seleccionado el parámetro puesto para el campo de busqueda1
          //Finalmente checa si la busqueda2 es por nombre, apellido o Id
          if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Nombre")
          Rs = St.executeQuery("CALL `ALL_PER_NAME_PUE`('"+busqueda1.getText()+"');");
          else
if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Apellido")
          Rs = St.executeQuery("CALL `ALL_PER_AP_PUE`('"+busqueda1.getText()+"');");
          else if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="ID")
          Rs = St.executeQuery("CALL `ALL_PER_ID_PUE`('"+busqueda1.getText()+"');");

          //búsqueda combinada para sucursal
          //Este caso es para cuando esta seleccionado el parámetro sucursal para el campo de busqueda1
          //Finalmente checa si la busqueda2 es por nombre, apellido o Id
          if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Nombre")
          Rs = St.executeQuery("CALL `ALL_PER_NAME_SUC`('"+busqueda1.getText()+"');");
          else
if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Apellido")
          Rs = St.executeQuery("CALL `ALL_PER_AP_SUC`('"+busqueda1.getText()+"');");
          else if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="ID")
          Rs = St.executeQuery("CALL `ALL_PER_ID_SUC`('"+busqueda1.getText()+"');");

          //búsqueda combinada para estado
          //Este caso es para cuando esta seleccionado el parámetro estado para el campo de busqueda1
          //Finalmente checa si la busqueda2 es por nombre, apellido o Id
          if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Nombre")
          Rs = St.executeQuery("CALL `ALL_PER_NAME_EST`('"+busqueda1.getText()+"');");
          else
if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Apellido")
          Rs = St.executeQuery("CALL `ALL_PER_AP_EST`('"+busqueda1.getText()+"');");
          else if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="ID")
          Rs = St.executeQuery("CALL `ALL_PER_ID_EST`('"+busqueda1.getText()+"');");
        }
        //si esta seleccionada la casilla activos pero no la de inactivos
        else if(act==true&&inact==false){
          //búsqueda combinada para puesto
          //Este caso es para cuando esta seleccionado el parámetro puesto para el campo de busqueda1
          //Finalmente checa si la busqueda2 es por nombre, apellido o Id
          if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Nombre")
          Rs = St.executeQuery("CALL `ALL_PER_NAME_PUE_ACT`('"+busqueda1.getText()+"');");
          else
if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Apellido")
          Rs = St.executeQuery("CALL `ALL_PER_AP_PUE_ACT`('"+busqueda1.getText()+"');");
          else if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="ID")
          Rs = St.executeQuery("CALL `ALL_PER_ID_PUE_ACT`('"+busqueda1.getText()+"');");

          //búsqueda combinada para sucursal
          //Este caso es para cuando esta seleccionado el parámetro sucursal para el campo de busqueda1
          //Finalmente checa si la busqueda2 es por nombre, apellido o Id
          if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Nombre")
          Rs = St.executeQuery("CALL `ALL_PER_NAME_SUC_ACT`('"+busqueda1.getText()+"');");
```

```
                else
if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Apellido")
                Rs = St.executeQuery("CALL `ALL_PER_AP_SUC_ACT`('"+busqueda1.getText()+"');");
                else if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="ID")
                Rs = St.executeQuery("CALL `ALL_PER_ID_SUC_ACT`('"+busqueda1.getText()+"');");

                //búsqueda combinada para estado
                //Este caso es para cuando esta seleccionado el parámetro estado para el campo de busqueda1
                //Finalmente checa si la busqueda2 es por nombre, apellido o Id
                if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Nombre")
                Rs = St.executeQuery("CALL `ALL_PER_NAME_EST_ACT`('"+busqueda1.getText()+"');");
                else
if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Apellido")
                Rs = St.executeQuery("CALL `ALL_PER_AP_EST_ACT`('"+busqueda1.getText()+"');");
                else if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="ID")
                Rs = St.executeQuery("CALL `ALL_PER_ID_EST_ACT`('"+busqueda1.getText()+"');");
              }
             //si esta seleccionada la casilla inactivos pero no la de activos
              else if(act==false&&inact==true){
                //búsqueda combinada para puesto
                //Este caso es para cuando esta seleccionado el parámetro puesto para el campo de busqueda1
                //Finalmente checa si la busqueda2 es por nombre, apellido o Id
                if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Nombre")
                 Rs = St.executeQuery("CALL `ALL_PER_NAME_PUE_INACT`('"+busqueda1.getText()+"');");
                else
if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="Apellido")
                Rs = St.executeQuery("CALL `ALL_PER_AP_PUE_INACT`('"+busqueda1.getText()+"');");
                else if(ComboBusqueda1.getSelectedItem()=="Puesto"&&ComboBusqueda2.getSelectedItem()=="ID")
                 Rs = St.executeQuery("CALL `ALL_PER_ID_PUE_INACT`('"+busqueda1.getText()+"');");

                //búsqueda combinada para sucursal
                //Este caso es para cuando esta seleccionado el parámetro sucursal para el campo de busqueda1
                //Finalmente checa si la busqueda2 es por nombre, apellido o Id
                if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Nombre")
                Rs = St.executeQuery("CALL `ALL_PER_NAME_SUC_INACT`('"+busqueda1.getText()+"');");
                else
if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="Apellido")
                Rs = St.executeQuery("CALL `ALL_PER_AP_SUC_INACT`('"+busqueda1.getText()+"');");
                else if(ComboBusqueda1.getSelectedItem()=="Sucursal"&&ComboBusqueda2.getSelectedItem()=="ID")
                Rs = St.executeQuery("CALL `ALL_PER_ID_SUC_INACT`('"+busqueda1.getText()+"');");

                //búsqueda combinada para estado
                //Este caso es para cuando esta seleccionado el parámetro estado para el campo de busqueda1
                //Finalmente checa si la busqueda2 es por nombre, apellido o Id
                if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Nombre")
                Rs = St.executeQuery("CALL `ALL_PER_NAME_EST_INACT`('"+busqueda1.getText()+"');");
                else
if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="Apellido")
                Rs = St.executeQuery("CALL `ALL_PER_AP_EST_INACT`('"+busqueda1.getText()+"');");
                else if(ComboBusqueda1.getSelectedItem()=="Estado"&&ComboBusqueda2.getSelectedItem()=="ID")
                Rs = St.executeQuery("CALL `ALL_PER_ID_EST_INACT`('"+busqueda1.getText()+"');");
              }
        } catch (Exception e) {
            //si hay algún error aquí se muestra
            JOptionPane.showMessageDialog(null,e);
        }
    }
```

Botón Modificar Personal.



La función del botón Modificar Personal manda a llamar al método de "modificarActionPerformed" cuando se la clic sobre él, este método realiza la modificación de los datos de un personal consultado anteriormente, primero obtiene la colonia del personal para llenar un ComboBox el cual será esta la primera opción, revisa si algunos de los campos donde se tiene valores numéricos este vacío, después realiza lo mismo para todos los demás campos que se tiene si se encuentra algún campo vacío mostrara un mensaje de "Campos vacíos", realiza búsquedas para encontrar datos los cuales tengan espacios de más, si encontró alguno mostrara un mensaje de dialogo de "Hay espacios en blanco de más" sino se entró a alguna de las decisiones anteriores se mostrara un mensaje de que si estas de modificar al personal, si selecciona que si entonces se realiza la conexión a la base de datos, se realiza una consulta para revisar si no hay ya un registro en la base de datos con el mismo nombre y apellidos, si no hay un registro igual entonces se llamara al método "AuxiliarModificar" en este método se obtiene el valor del id del puesto y sucursal, se obtienen todos los valores ingresados en los campos de texto y posteriormente se realiza un "update" del personal actualizado y se manda a llamar al método "Cargardatos tabla" en el cual se cargan todo los datos de todos los registros de personal se notifica que el personal se ha modificado exitosamente y se llama al método de "limpiacampos" este método quita todos los datos ingresados en los campos de texto dejándolos en blanco, después se regresa al método "modificarActionPerformed" se cierran las conexiones abiertas y finaliza el método.

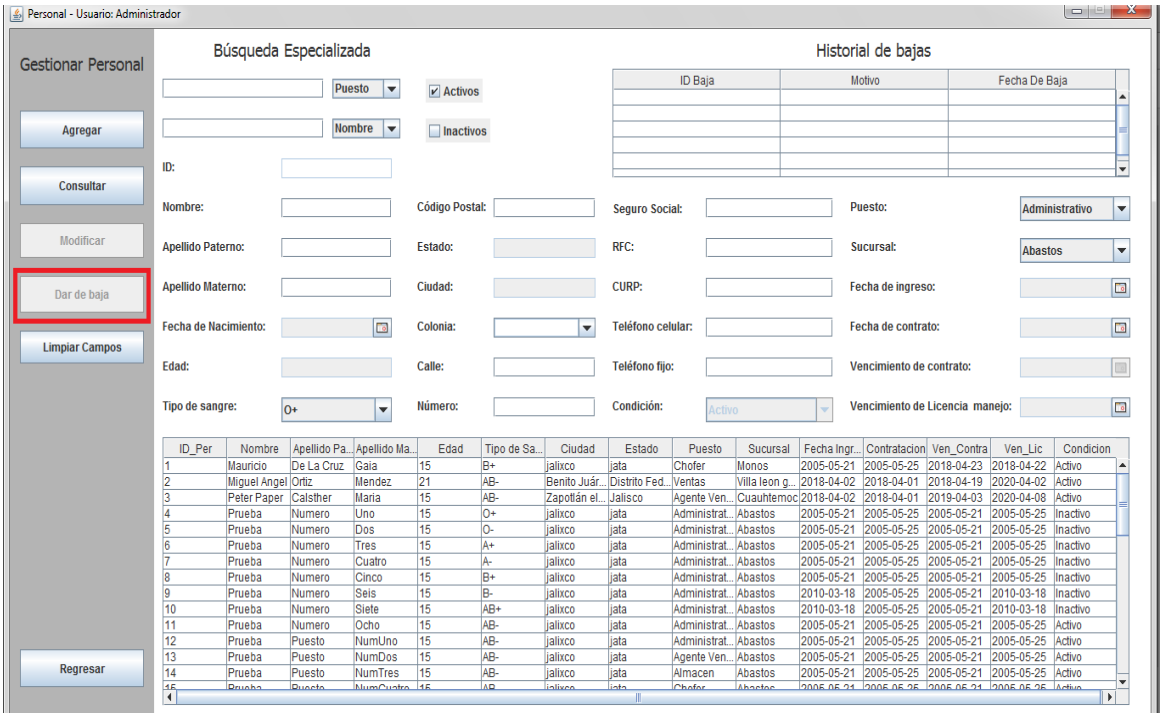
Código:

```
private void modificarActionPerformed(java.awt.event.ActionEvent evt) {
        //Aquí se definen las banderas y auxiliares que se utilizaran
        int auxedad=0,auxcp=0;
        ColCombobox=(String)colonia.getSelectedItem();

        //Se comprueba si hay campos vacíos en números
        if(edad.getText().isEmpty())auxedad=1;
        else {int EDAD=Integer.parseInt(edad.getText());}
        if(cp.getText().isEmpty())auxcp=1;
        else {int CP=Integer.parseInt(cp.getText());}

 //Se revisa si hay campos vacíos que no sean de tipo entero
if(nom.getText().isEmpty()||ap.getText().isEmpty()||am.getText().isEmpty()||df.format(FechaNa.getDate()).isEmpty()||call
e.getText().isEmpty()||numero.getText().isEmpty()||ciudad.getText().isEmpty()||estado.getText().isEmpty()||nss.getText().i
sEmpty()||rfc.getText().isEmpty()||curp.getText().isEmpty()||celular.getText().isEmpty()||fijo.getText().isEmpty()||df.forma
t(FechaIngre.getDate()).isEmpty()||df.format(FechaContra.getDate()).isEmpty()||df.format(FechaVenContra.getDate()).isE
mpty()||df.format(FechaVenLM.getDate()).isEmpty()||auxedad==1||auxcp==1||ColCombobox.isEmpty()){

       //Si encontró alguno entonces mostrara este mensaje
        JOptionPane.showMessageDialog(null,"Campos vacios");
       }//if (validar vampos vacios)
       else{
          //validamos que no tenga espacios de más en los campos necesarios para la búsqueda
          boolean correcto=true;
          if(String.valueOf(nom.getText().charAt(0)).equals("
")||String.valueOf(nom.getText().charAt(nom.getText().length() - 1)).equals(" "))
          {
            labelnombre.setForeground(Color.red);correcto=false;
          }
          if(String.valueOf(ap.getText().charAt(0)).equals(" ")||String.valueOf(ap.getText().charAt(ap.getText().length() -
1)).equals(" "))
          {
            labelap.setForeground(Color.red);correcto=false;
          }
          if(String.valueOf(am.getText().charAt(0)).equals(" ")||String.valueOf(am.getText().charAt(am.getText().length() -
1)).equals(" "))
          {
            labelam.setForeground(Color.red);correcto=false;
          }
          if(String.valueOf(estado.getText().charAt(0)).equals("
")||String.valueOf(estado.getText().charAt(estado.getText().length() - 1)).equals(" "))
          {
            labelestado.setForeground(Color.red);correcto=false;
          }
          if(correcto==false){
//Se notifica al usuario que tiene espacios en blanco de más
            JOptionPane.showMessageDialog(null,"Hay espacios en blanco de mas");
          }
//Sino encontró algún campo con espacios de más entonces mostrara el siguiente mensaje de selección con SI,NO
          else{
           Object [] opciones ={"Aceptar","Cancelar"};
           int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estás seguro de modificar este registro?","Mensaje de
Confirmacion",
           JOptionPane.YES_NO_OPTION,
           JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
           //Si se selecciona Si entonces realizara una conexión a la base de datos
           if (eleccion == JOptionPane.YES_OPTION)
           {
              try{
             //Se realiza la conexión a la base de datos
```

```
                Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
                St=Con.createStatement();

                //sacamos el actual nombre del id que se muestra a partir de su nombre completo
                //Rs = St.executeQuery("SELECT  CONCAT(nom_p,' ',ap,' ',am) as Repito FROM personal WHERE ID_Per
= "+Integer.parseInt(id.getText())+";");
                Rs = St.executeQuery("CALL BNPI("'+Integer.parseInt(id.getText())+"');");
                while(Rs.next()){ConsultaNombre=Rs.getString("Repito");}

                //ahora sacamos el nuevo nombre que modificara
                String nombre=(nom.getText()+" "+ap.getText()+" "+am.getText());

                //ahora podemos ver que al modificar el registro no le ponga nombre y apellidos iguales a otro existente en la
base de datos
                if(nombre.equals(ConsultaNombre)){
                 //JOptionPane.showMessageDialog(null,"Son iguales");
                 AuxiliarModificar();//el registro es el que originalmente se consultó, así que si está en la base de datos y
omitimos validar su existencia
                 //Se desactivan los botones de modificar y dar de baja
                 modificar.setEnabled(false);
                 baja.setEnabled(false);
                }
                else
                {
                Rs = St.executeQuery("SELECT  CONCAT(nom_p,' ',ap,' ',am) as Repito FROM personal WHERE
CONCAT(nom_p,' ',ap,' ',am) = "'+nombre+"';");
                   //Si lo encontró entonces mostrara el siguiente mensaje
                  if(Rs.last()){
                       JOptionPane.showMessageDialog(null,"Ya existe un registro con ese nombre");
                    }
                    else{
                       //Sino mandara a llamar a método de AuxiliarModificar
                        AuxiliarModificar();
                        //Se desactivan los botones de modificar y dar de baja
                        modificar.setEnabled(false);
                        baja.setEnabled(false);
                    }//else de si existe ya el registro o no en la base de datos
                 }//else de si el registro es el mismo que originalmente se consulto
                //Cierra la conexión a la base de datos
                Con.close();
                Rs.close();
                St.close();
                SeleccionCombo2();//actualiza la búsqueda especializada
              } catch (Exception e){
                 JOptionPane.showMessageDialog(null,e);
              }
          }//if de confirmation
        }//else de espacios de mas
      }//else de campos vacíos
    }
     //Este método se llama cada vez que se realiza una modificación
    public void AuxiliarModificar(){
       //Se declaran las variables a utilizar
       String nacimiento,ingreso,contra,ven_contra,ven_lm;
       int idpue=0,idsuc=0;
       try {
                 //necesitamos encontrar el id de puesto y sucursal con el nombre elegido
                 Rs=St.executeQuery("select ID_Pue from puesto where nom_p = "'+P+"';");
                 while(Rs.next())idpue=Integer.parseInt(Rs.getString("ID_Pue"));
                 Rs=St.executeQuery("select ID_Suc from sucursal where nom_suc = "'+S+"';");
                 while(Rs.next())idsuc=Integer.parseInt(Rs.getString("ID_Suc"));
```

```
                //obtenemos los formatos correctos de las fechas
                nacimiento=GetFormatoFecha("FechaNa");
                ingreso=GetFormatoFecha("FechaIngre");
                contra=GetFormatoFecha("FechaContra");//de la tabla contratos
                ven_contra=GetFormatoFecha("FechaVenContra");//de la tabla contratos
                ven_lm=GetFormatoFecha("FechaVenLM");

                //obtenemos la condicion
                String condi=(String)condicion.getSelectedItem();

               //ahora si a modificar en la tabla personal
                String query="update Personal set ID_Pue = "+idpue+",ID_Suc =
"+idsuc+",nom_p='"+nom.getText()+"',ap='"+ap.getText()+"',am='"+am.getText()+"',Fecha_Nacimiento='"+nacimiento+
"',edad='"+edad.getText()+"',tipo_s='"+TPS+"',calle='"+calle.getText()+"',numero='"+numero.getText()+"',colonia='"+Co
lCombobox+"',cp='"+cp.getText()+"',ciudad='"+ciudad.getText()+"',estado='"+estado.getText()+"',nss='"+nss.getText()+"
',rfc='"+rfc.getText()+"',curp='"+curp.getText()+"',cel='"+celular.getText()+"',fijo='"+fijo.getText()+"',Fecha_Ingreso='"+
ingreso+"',Fecha_Contrato='"+contra+"',Vencimiento_Contrato='"+ven_contra+"',Vencimiento_Licencia='"+ven_lm+"',c
ondicion= '"+condi+"' where ID_Per = "+Integer.parseInt(id.getText())+";";
                St.executeUpdate(query);//modificacion del personal

                //cargardatosTabla("select * from Personal");
                cargardatosTabla("CALL CDTP");
                //Se notifica que el registro se modificó exitosamente
                JOptionPane.showMessageDialog(null,"Registro modificado exitosamente");
                //Se manda a llamar al metodo limpiar campos
                                                limpiacampos();
                 //Se remueven todos los ítems de colonia en su autocompleter
                colonia.removeAllItems();
          //Si ocurre alguna excepción esta se atrapa aquí
        } catch (Exception e) {
          JOptionPane.showMessageDialog(null,e);
        }
    }
```

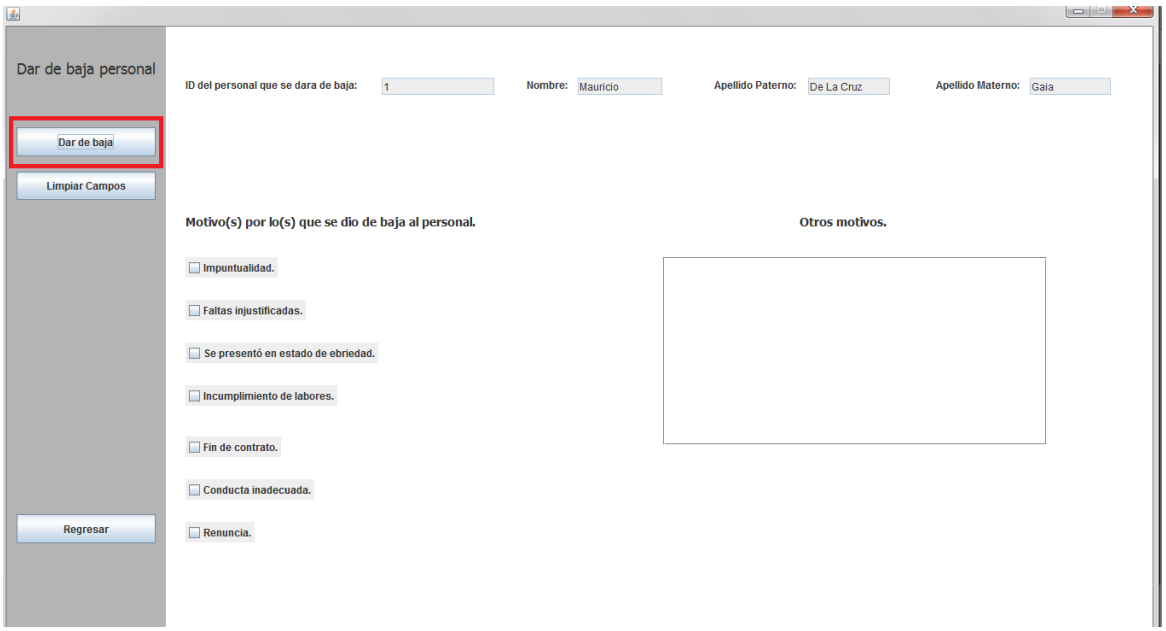
Botón Dar de Baja



La función del botón Dar de baja es de llamar a llamar al método “bajaActionPerformed” el cual abre una nueva ventana cerrando la actual y mandando los siguientes parámetros al frame a visualizar: identificador del personal, nombre y sus apellidos.

Código

```
private void bajaActionPerformed(java.awt.event.ActionEvent evt) {
        //Abre la ventana de DarBaja
        new DarBaja().setVisible(true);
        //Cierra la ventana actual
        this.dispose();
        //Manda los siguientes valores a la ventana que se abrió para su utilización.
        DarBaja.idb.setText(id.getText());
        DarBaja.nombaja.setText(nom.getText());
        DarBaja.apbaja.setText(ap.getText());
        DarBaja.ambaja.setText(am.getText());
    }
```



Esta ventana se abre cuando se selecciona dar de baja en la interfaz de “Gestionar personal” el botón de Dar de baja tiene como función revisar si hay campos de texto vacíos, de encontrarlos mostrara el mensaje de “Campos vacíos” de no encontrarlos entonces establecerá conexión con la base de datos, realizara la una consulta para obtener la fecha actual del sistema, comenzara a concatenar en una cadena llamada “Mot” los motivos de la baja que se ingresaron por medio de las casillas de verificación, se realiza un insert a la base de datos en cual se envían los datos del personal dado de baja y los motivos ingresados en las casillas de selección y en el text área, se realiza un update del personal dado de baja actualizando su estado a “Inactivo” se limpian los campos de texto, se muestra un mensaje de personal dado de baja exitosamente, se cierra la ventana actual y se muestra la interfaz de “Gestionar personal”. Si se utiliza limpiar campos esta manda a llamar al método llamado

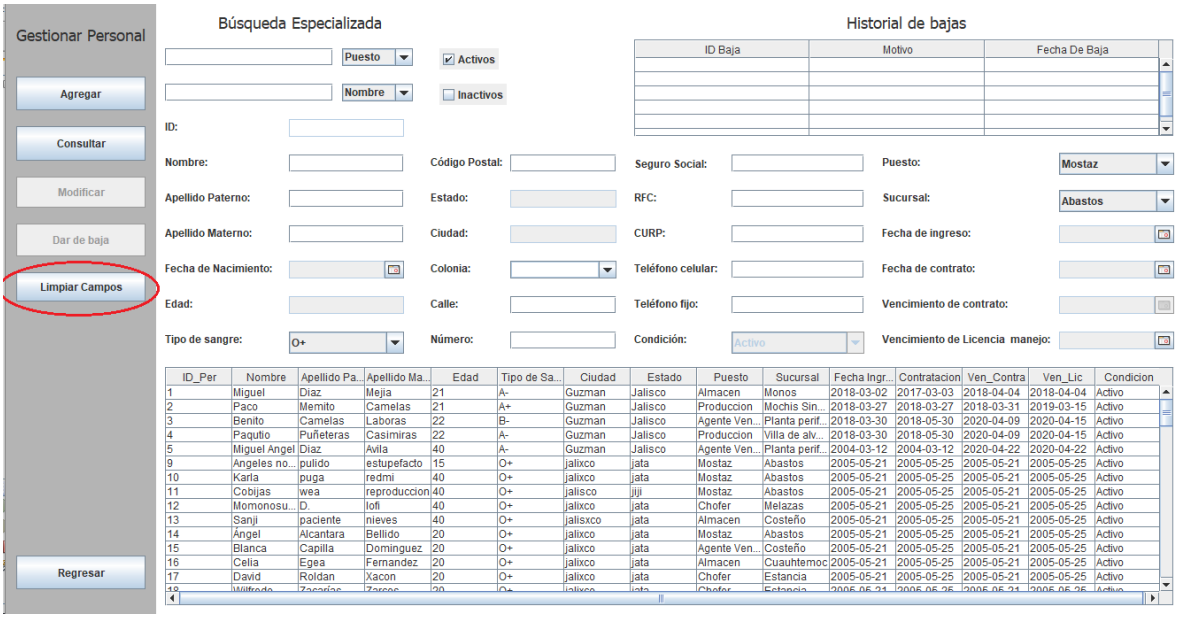
“LPActionPerformed” el cual deselecciona todas las casillas de verificación y deja vacío el text área.

Código

```
private void darbajaActionPerformed(java.awt.event.ActionEvent evt) {
 //Revisa si hay campos vacíos
 if(motivo.getText().isEmpty() &&
(vcbee==0&&vcbi==0&&vcbfi==0&&vcbil==0&&vcbfc==0&&vcbci==0&&vcbr==0))JOptionPane.showMessageDial
og(null,"Campos vacios");//Si los encuentra muestra el este mensaje
       else{
       try {
//Sino realiza conexión con la base de datos.
Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
            St=Con.createStatement();
//Si inicializa la cadena Hoy en vacío.
            String Hoy="";
//Se realiza una consulta obteniendo la fecha actual del sistema
            Rs=St.executeQuery("select CURDATE() as Fecha_Baja;");
//Se obtienen los motivos de las Casillas de verificación seleccionadas
            while(Rs.next())Hoy=(Rs.getString("Fecha_Baja"));
            if(imp.isSelected()){Mot=Mot+" \\nImpuntualidad ";}
            if(fali.isSelected()){Mot=Mot+" \\nFaltas Injustificadas ";}
            if(ebri.isSelected()){Mot=Mot+" \\nSe presento en estado de ebriedad ";}
            if(inla.isSelected()){Mot=Mot+" \\nIncumplimiento de labores ";}
            if(fincontra.isSelected()){Mot=Mot+" \\nFin de contrato ";}
            if(malacon.isSelected()){Mot=Mot+" \\nMala conducta ";}
            if(renuncia.isSelected()){Mot=Mot+" \\nRenuncia ";}
//Se realiza el inserte en la base de datos con los datos ingresados anteriormente
            String query="insert into Bajas(ID_Per,motivo,Fecha_Baja)
values('"+idb.getText()+"','"+motivo.getText()+Mot+"','"+Hoy+"');";
                St.executeUpdate(query);
//Se actualiza el estado del personal a inactivo
            query="update Personal set condicion= 'Inactivo' where ID_Per = "+Integer.parseInt(idb.getText())+";";
                St.executeUpdate(query);
//Se ponen todos los campos de texto, Casillas de verificación y el text area en vacíos.
                idb.setText("");
                nombaja.setText("");
                apbaja.setText("");
                ambaja.setText("");
                motivo.setText("");
                imp.setSelected(false);
                fali.setSelected(false);
                ebri.setSelected(false);
                inla.setSelected(false);
                fincontra.setSelected(false);
                malacon.setSelected(false);
                renuncia.setSelected(false);

//Se muestra mensaje de registro dado de baja exitosamente.
                JOptionPane.showMessageDialog(null,"Registo dado de baja exitosamente");
//Se cierra la ventana actual
                this.dispose();
//Se abre interfaz de Gestionar personal
                new Personal().setVisible(true);
       } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null,ex);
           Mot="";
       }
      }//else campos vacios
```

```
}


        private void LPActionPerformed(java.awt.event.ActionEvent evt) {
    //Se ponen todos los campos de texto, Casillas de verificación y el text área en vacíos.
    motivo.setText("");
    Mot="";
    imp.setSelected(false);
    fali.setSelected(false);
    ebri.setSelected(false);
    inla.setSelected(false);
    fincontra.setSelected(false);
    malacon.setSelected(false);
    renuncia.setSelected(false);
}
```

Botón Limpiar campos.



Método LPActionPerformed

El botón limpiar campos tiene un ActionPerformed el cual llama al método LimpiarCampos.

```
private void LPActionPerformed(java.awt.event.ActionEvent evt) {   //Botón "Limpiar  Campos"
  limpiacampos(); }   //Se invoca el método limpiar campos
```

Método limpiacampos

El método "limpiarcampos" manda a todos los campos de texto los valores vacíos, desactiva los botones de modificar, y dar de baja, remueve todos los ítems que tiene el campo colonia, manda al método cargardatosTabla el procedimiento "CDTP", se mueve el cursos al campo de texto "nom", los campos de FechaNa, FechaIngre,FechaContra, FechaVencontra, FechaVenLM, se cambia su valor por "null", el campo FechaVenContra se desactiva, el combobox condición se selecciona la opción de "Activo", y se desactiva el combobox de condición, se llama al método "limpiartabla2" y se activan los check boton de activos e inactivos.

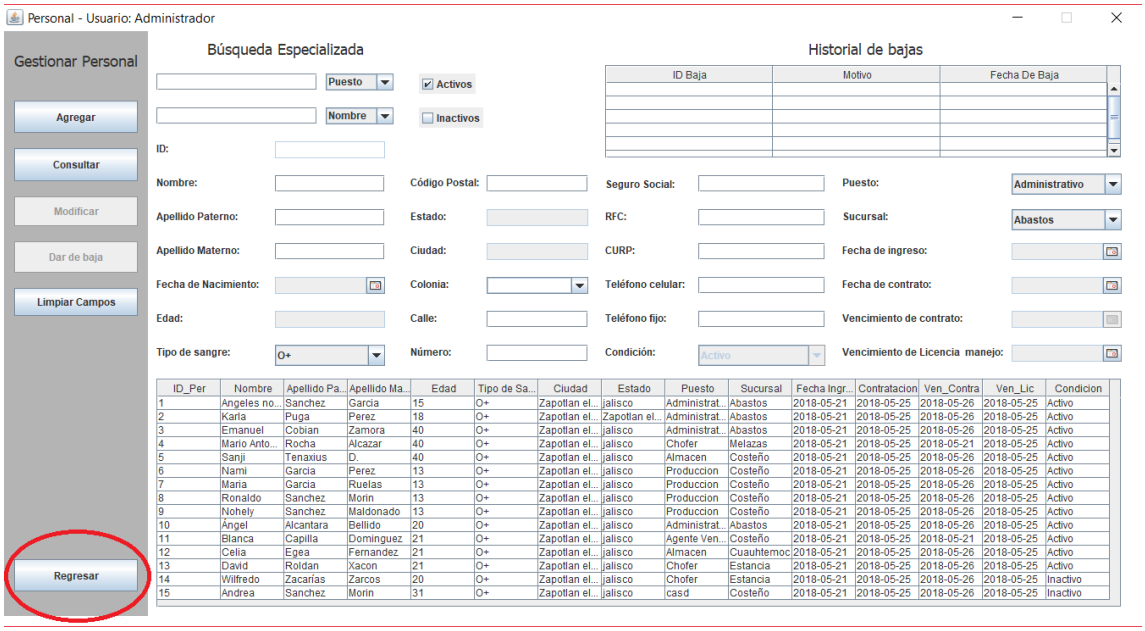
```
public void limpiacampos(){
        id.setText("");
        nom.setText("");
        ap.setText("");
        am.setText("");
        edad.setText("");
        calle.setText("");
        numero.setText("");
        colonia.removeAllItems();
        cp.setText("");
        ciudad.setText("");
        estado.setText("");
        nss.setText("");
        rfc.setText("");
        curp.setText("");
        celular.setText("");
        fijo.setText("");
        modificar.setEnabled(false);
        baja.setEnabled(false);
        //cargardatosTabla("select*from Personal");
        cargardatosTabla("CDTP");
        nom.requestFocus();
        busqueda1.setText("");
        busqueda2.setText("");
        FechaNa.setCalendar(null);
        FechaIngre.setCalendar(null);
        FechaContra.setCalendar(null);
        FechaVenContra.setCalendar(null);
        FechaVenContra.setEnabled(false); //se desactiva la fecha de vencimiento de contrato
        FechaVenLM.setCalendar(null);
        condicion.setSelectedItem("Activo");
        condicion.setEnabled(false);
        limpiartabla2();
        activos.setSelected(true);
        inactivos.setSelected(true);
        SeleccionCombo2();//vuelve a rellenar el autocompleter de la busqueda 2 con lo que este seleccionado en el
combobox 2, ya sean activos o inactivos
        labelnombre.setForeground(Color.black);
        labelap.setForeground(Color.black);
        labelam.setForeground(Color.black);
        labelestado.setForeground(Color.black);
        labelcelular.setForeground(Color.black);
        labelfijo.setForeground(Color.black);
}
```

Método limpiartabla2

El "Método limpiartabla2" vacía la tabla que contiene historial de un personal que tiene bajas dentro del sistema, el método recorre todas las celdas y remueve los datos.

```
public void limpiartabla2(){
   DefaultTableModel tb = (DefaultTableModel) jTable2.getModel();
   for (int x = 0; x < jTable2.getRowCount(); x++) {
      tb.removeRow(x);
      x-=1;
   }
}
```

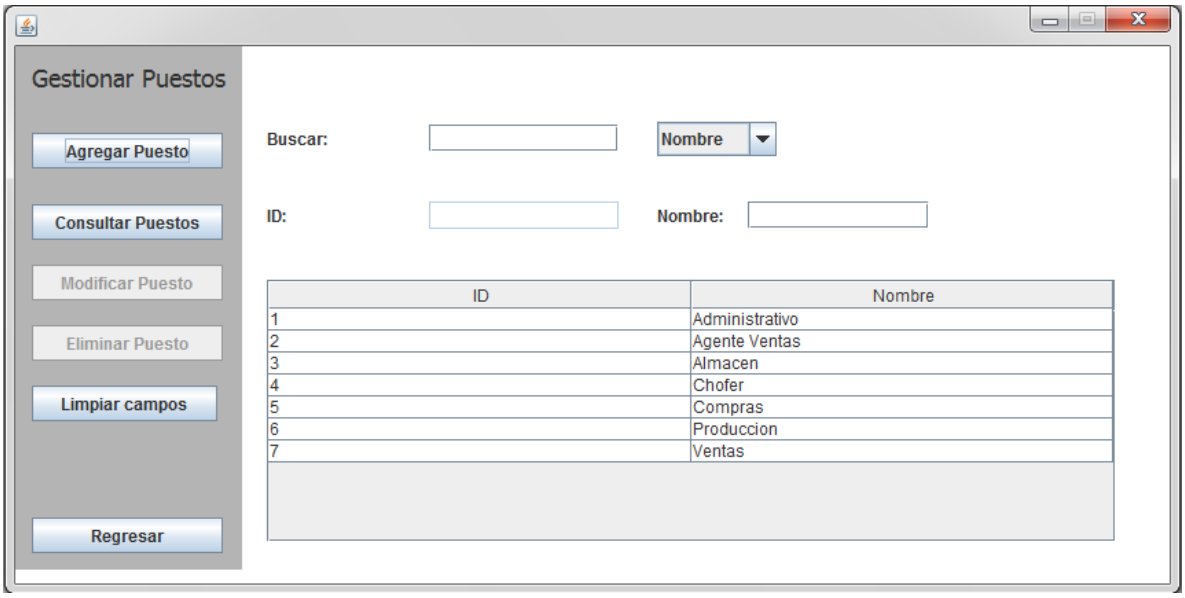
Botón Regresar Personal



Al presionar el botón “Regresar” se manda a llamar a su método regresarActionPerformed el cual se encarga de cerrar las ventas de alerta si es que estas están abiertas, cierra la misma ventana del módulo personal y finalmente abre el menú principal.

```
private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
        avc.dispose();//se cierra la alerta contrato
        avl.dispose();//se cierra la alerta licencia
        this.dispose();//se cierra esta ventana
        new ControlPanel().setVisible(true);//se abre el panel de control
    }
```

Modulo Gestionar Puestos.



En este módulo se utilizan las siguientes librerías todas son requeridas para el correcto funcionamiento de todas las funciones:

```
//Importación de las librerías necesarias para el módulo
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import com.mxrck.autocompleter.AutoCompleterCallback;
import com.mxrck.autocompleter.TextAutoCompleter;
import java.awt.event.KeyEvent;
```

Este módulo también hace uso de las siguientes variables globales que se utilizaran en las funciones que se tiene.

```
//Se declaran las variables globales que se utilizaran en las funciones y métodos del sistema.
   Connection Con; //Conexion
    Statement St;   //Ejecutar comandos
    ResultSet Rs;   //Resultado de la consulta
    static int x=0;
    TextAutoCompleter autocompleta; //Para autocompletar los campos requeridos
```

El modulo utiliza los siguientes métodos que son utilizados por la mayoría de funciones para su correcto funcionamiento.

Método llenacampos

```
public void llenacampos(String parametro){
            //Se revisa que el parámetro recibido no este vacío
     if(parametro.isEmpty()){
       JOptionPane.showMessageDialog(null,"Campo de busqueda vacio");//Si esta vacío entonces mostrara este mensaje
      }
      else{//Sino continua aquí
         try {
          //Establece conexión con la base de datos
          Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
          St=Con.createStatement();
          String query="";
         //Se declara una variable del tipo booleano
         boolean existe;
                        //Dependiendo de la manera a buscar a partir del ComboBox para las búsquedas
         if(ComboBusqueda.getSelectedItem()=="Nombre"){
           Rs = St.executeQuery("CALL BNOMPue('"+parametro+"');");//Si es por nombre se realiza la siguiente consulta
         }
         else if(ComboBusqueda.getSelectedItem()=="ID"){//Si es por ID
           parametro=quitaEspacios(parametro);//quitamos el espacio de más que se agregó al rellenar el autocompleter
llamando al método quitaEspacios
           Rs = St.executeQuery("CALL BIDPue("+Integer.parseInt(parametro)+");");  //Realiza la siguiente consulta
         }
             //a continuación nos damos cuenta si existe o no
              if(Rs.last()){existe=true;}
              else{
                JOptionPane.showMessageDialog(null,"No existe el registro"); //Sino existe el registro muestra este mensaje
                existe=false; //Pone la variable booleana creada antes en false
              }

            if(existe==true){ //Si lo encontró entonces continua
                if(ComboBusqueda.getSelectedItem()=="ID"){//Revisa si la búsqueda es por ID
                    query="CALL BIDPue("+Integer.parseInt(parametro)+");"; //Si lo es realiza esta consulta
                }
                else if(ComboBusqueda.getSelectedItem()=="Nombre"){//Sino revisa si es por nombre
                    query="CALL BNOMPue('"+parametro+"');";//Si lo es entonces realiza esta consulta
                }
                Rs=St.executeQuery(query);//Ejecuta la consulta
                cargardatos(); //Se llama al método cargardatos
                modificar.setEnabled(true); //Se activa el botón de modificar
                eliminar.setEnabled(true); //Se activa el botón de eliminar
                busqueda.setText(""); //Se limpia el campo de búsqueda
          }

         //Se cierra conexión con la base de datos
         St.close();
         Rs.close();
         Con.close();

       } catch (Exception e) { //Si llega a ocurrir una excepción se atrapara aquí
         JOptionPane.showMessageDialog(null,e);
       }
      }
   }
```

Método quitaEspacios

```
//Este método elimina los espacios de más de una cadena mandada como parámetro de entrada.
public String quitaEspacios(String texto) {
        java.util.StringTokenizer tokens = new java.util.StringTokenizer(texto);
        StringBuilder buff = new StringBuilder();
        while (tokens.hasMoreTokens()) {
            buff.append(" ").append(tokens.nextToken());
        }
        return buff.toString().trim();
    }
```

Método ComboBusquedaActionPerformed

```
        //Este método define la manera en la que se va a realizar la búsqueda de los puestos ya sea por id o por nombre
            private void ComboBusquedaActionPerformed(java.awt.event.ActionEvent evt) {
        x=ComboBusqueda.getSelectedIndex();//Se inicializa x con el valor que se tiene en la tabla por default
        if(x==0 || x==1){//Revisa si hay seleccionado algún valor en el combobox
            busqueda.setText("");//Pone el campo de búsqueda vacío.
            //cargardatosTabla("select*from Puesto");
            cargardatosTabla("call TODOPue"); //Manda a llamar al método de cargardatosTabla
        }
        autocompleta.removeAllItems();//Remueve todas las opciones en el autocompleta
        autocompletaDatos(); //Llama al método autocompletaDatos para cargar los datos en el autocompleter
    }
```

Método cargardatosTabla

```
//Este método carga los datos del todos los puestos a una tabla
public void cargardatosTabla(String consulta){
        try {
            DefaultTableModel tabla=new DefaultTableModel(); //Se crea una nueva tabla
            //Se establece conexión con la base de datos
            Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
            St=Con.createStatement();
            Rs=St.executeQuery(consulta);//Realiza la consulta recibida como parámetro de entrada
            // Se ingresan los identificadores de la tabla
            tabla.setColumnIdentifiers(new Object[]{"ID","Nombre"});
            //mientras obtenga resultados de la búsqueda los ira poniendo en la tabla
            while(Rs.next()){
             tabla.addRow(new Object[]{Rs.getString(1),Rs.getString(2)});
            }
            jTable1.setModel(tabla);//Se crea la tabla
            //Se cierra la conexión con la base de datos
            St.close();
            Rs.close();
            Con.close();
        } catch (Exception e) { //Si ocurre una excepción se atrapa aquí
            JOptionPane.showMessageDialog(null,e);
        }
    }
```

Método formWindowOpened

```
    //Este método se ejecuta automáticamente cuando se ingresa a la interfaz de Gestionar Puestos
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
   //cargardatosTabla("select*from puesto");
   //Llama al método cargar datos tabla con la consulta de cargar todos los datos de puestos
       cargardatosTabla("call TODOPue");
```

```
  //Llama al método de autocompletaDatos para cargar los datos en el autocompleter
      autocompletaDatos();
   }
```

Método ComboBusquedaActionPerformed

```
  //Este método se encarga de verificar si hay algo selecciona en el ComboBox y saber cuál de los ítems es
  private void ComboBusquedaActionPerformed(java.awt.event.ActionEvent evt) {
      //Se declara una variable la cual tendrá el valor decimal del ítem del ComboBox
                        x=ComboBusqueda.getSelectedIndex();
      //Si hay algo seleccionado entonces entra a esta condición
      if(x==0 || x==1){
         busqueda.setText("");
         //cargardatosTabla("select*from Puesto");
         //Llama al método cargardatosTabla con la consulta de cargar todo de puestos
         cargardatosTabla("call TODOPue");
      }
       //Remueve todos los ítems del autocompleta
      autocompleta.removeAllItems();
       //Los vuelve a cargar ya actualizados
      autocompletaDatos();
   }
```

Método cargardatos

```
//Este método carga los datos donde se solicite mientras encuentre registros
 public void cargardatos(){
        try {
//Mientras encuentre un registro siguiente al que está actualmente se repetirá
          while(Rs.next()){
//Los datos encontrados los pondrá en su respectivo campos el cual es ID y Nombre en este caso
           id.setText(Rs.getString(1));
           nom.setText(Rs.getString(2));
          }
//Si ocurre alguna excepción se atrapara aquí
       } catch (Exception e) {
          JOptionPane.showMessageDialog(null,"Error"+e);
       }
}
```

Método nomKeyTyped

```
//se manda a llamar a este método automáticamente para validar que sea un carácter correcto
private void nomKeyTyped(java.awt.event.KeyEvent evt) {
//primero se guarda en la cadena nombre lo que hay en el campo
//de texto antes del nuevo carácter ingresado
String nombre=nom.getText();
//se obtiene solo el carácter ingresado
char k=evt.getKeyChar();

//solo si el carácter ingresado es una letra, un punto o un espacio se
//dejara ingresar y se analizara el orden correcto para la validación
if(Character.isLetter(k)||k=='.'||k==' '){
//si el campo de texto no está vacío
if(nombre.length()>0){
//si el último carácter en el campo de texto es un punto
//y el carácter ingresado es diferente de un espacio
//entonces no se deja ingresar este carácter
if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
getToolkit().beep();
evt.consume();
}
```

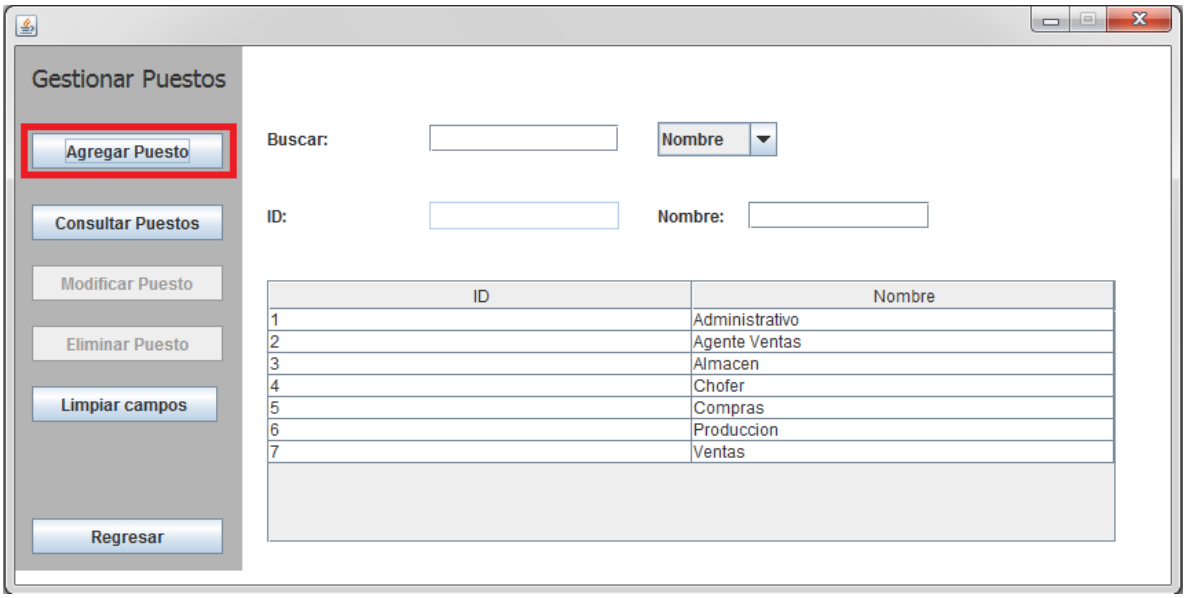
```
//si el último carácter en el campo de texto es un espacio
//y el carácter ingresado es un espacio o un punto
////entonces no se deja ingresar este carácter
else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
getToolkit().beep();
evt.consume();
}
//no se deja ingresar un carácter si en el campo de texto ya hay 14 caracteres
else if(nombre.length()==14){
getToolkit().beep();
evt.consume();
}
}
//si el campo de texto si está vacío no se puede
//ingresar espacios o puntos
else if(nombre.isEmpty()){
if(k==' '||k=='.'){
getToolkit().beep();
evt.consume();
}
}
}
//si el carácter es algo diferente de una letra, punto o espacio
//no se deja ingresar el carácter
else{
evt.consume();
}
//si el campo de texto ya contiene un punto ya no deja ingresar otro
if(nombre.contains(".")&&k=='.'){
getToolkit().beep();
evt.consume();
}
}
```

Método busquedaKeyTyped

```
//se manda a llamar a este método automáticamente para validar que sea un carácter correcto
 private void busquedaKeyTyped(java.awt.event.KeyEvent evt) {
       //Se revisa si se tiene seleccionado nombre en el ComboBox
       if(ComboBusqueda.getSelectedItem()=="Nombre"){
        //primero se guarda en la cadena nombre lo que hay en el campo de texto antes del nuevo carácter ingresado
        String nombre=busqueda.getText();
         //se obtiene solo el carácter ingresado
         char k=evt.getKeyChar();
        //solo si el carácter ingresado es una letra, un punto o un espacio se
        //dejara ingresar y se analizara el orden correcto para la validación
        if(Character.isLetter(k)||k=='.'||k==' '){
         //si el campo de texto no está vacío
         if(nombre.length()>0){
           //si el último carácter en el campo de texto es un punto
           //y el carácter ingresado es diferente de un espacio
           //entonces no se deja ingresar este carácter
           if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
           getToolkit().beep();
           evt.consume();
          }
          //si el último carácter en el campo de texto es un espacio
          //y el carácter ingresado es un espacio o un punto
          //entonces no se deja ingresar este carácter
          else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
           getToolkit().beep();
           evt.consume();
```

```
        }
         //no se deja ingresar un carácter si en el campo de texto ya hay 14 caracteres
         else if(nombre.length()==14){
           getToolkit().beep();
           evt.consume();
         }
        }
        //si el campo de texto si está vacío no se puede
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
            getToolkit().beep();
            evt.consume();
          }
        }
       }else{
       //si el carácter es algo diferente de una letra, punto o espacio
       //no se deja ingresar el carácter
           //getToolkit().beep();
           evt.consume();
       }
       //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
           getToolkit().beep();
           evt.consume();
        }
    }
   //Se comprueba si esta seleccionado el ID en el ComboBox
    else if(ComboBusqueda.getSelectedItem()=="ID"){

     //se obtiene solo el carácter ingresado
      char k=evt.getKeyChar();
      //primero se guarda en la cadena nombre lo que hay en el campo
      //de texto antes del nuevo carácter ingresado
      String ID=busqueda.getText();
      //Se revisa si la cadena está compuesta de dígitos
      if(Character.isDigit(k)){
      //Se comprueba si está vacía la cadena
         if(ID.length()>0){
          //De igual manera que no pase de una longitud de 10 dígitos
           if(ID.length()==10){
            //si el carácter es algo diferente de una letra, punto o espacio
            //no se deja ingresar el carácter
            //getToolkit().beep();
            evt.consume();
           }
          }
       }
      else{
        //si el carácter es algo diferente de una letra, punto o espacio
        //no se deja ingresar el carácter
        evt.consume();
       }//else
     }
    }
```

Botón Agregar Puesto

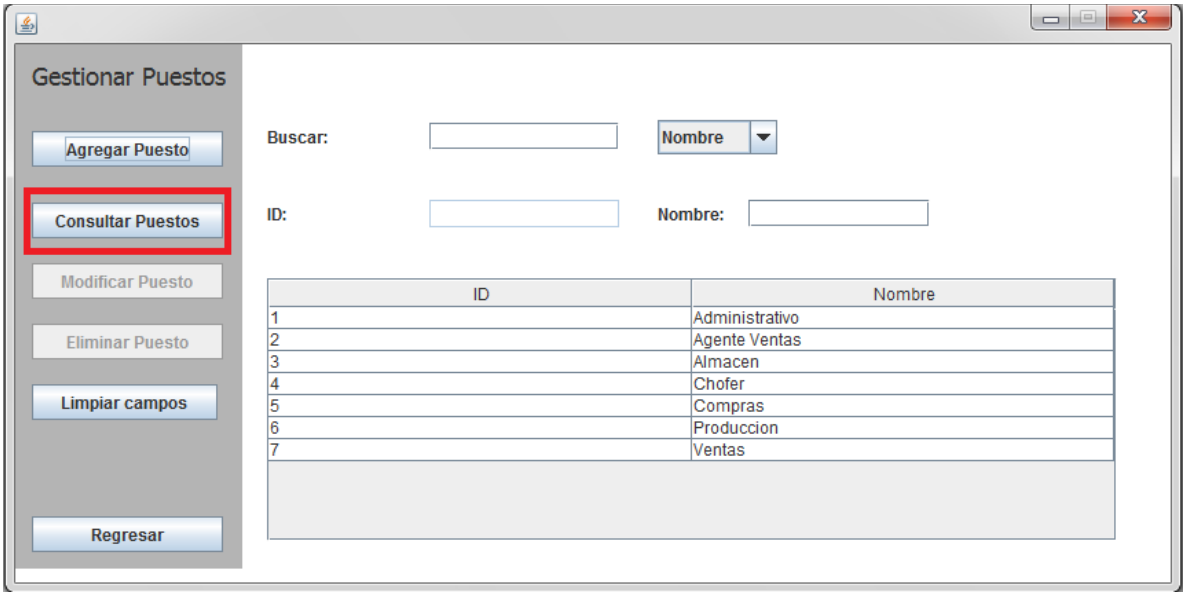


La función del botón agregar puesto es manda a llamar al método "agregarActionPerformed" cuando se dé clic en él, este método realiza la agregación de puesto a la base de datos del sistema de la siguiente manera: revisa que no se tengan campos vacíos si los hay entonces mostrara el mensaje de "Campos vacíos" de lo contrario continuara mostrando un mensaje de selección en el cual nos dirá si deseamos agregar un nuevo registro a la base de datos si seleccionamos que si entonces se establecerá una conexión a la base de datos del sistema, ejecutara una consulta para revisar sino existe un registro con los mismos datos que el que se desea agregar en caso de que lo encuentre se mostrara un mensaje diciendo que ya existe un registro con ese nombre en caso contrario se realiza una consulta donde se inserte el registro con la información proporcionada en los campos de texto, se le notificara al usuario que el registro se agregó correctamente, se cerrara la conexión con el sistema por último se desactivaran los botones de modificar y eliminar si estos estaban activos.

```
//Este método agrega un puesto a la base de datos
private void agregarActionPerformed(java.awt.event.ActionEvent evt) {
        //Revisa que no se tengan campos vacíos
        if(nom.getText().isEmpty()){JOptionPane.showMessageDialog(null,"Campos vacios");}//Si los hay mostrara el
siguiente mensaje
        else{
```

```
          Object [] opciones ={"Aceptar","Cancelar"};
         //Sino se mostrara el mensaje de selección
          int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estas seguro de agregar un nuevo registro?","Mensaje de Confirmacion",
          JOptionPane.YES_NO_OPTION,
          JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");

          if (eleccion == JOptionPane.YES_OPTION)
          {
              try {
             //Si se selecciona que si entonces se establece conexión con el servidor
              String nombre=nom.getText();
              Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
              St=Con.createStatement();
              //Se ejecuta la siguiente consulta
              Rs = St.executeQuery("SELECT nom_p FROM puesto WHERE nom_p = "'+nombre+'"");
              if(Rs.last()){
                 JOptionPane.showMessageDialog(null,"Ya existe un registro con ese nombre");//Si ya existe un registro con ese nombre muestra el siguiente mensaje
                 String iddsd=Rs.getString(1);
              }
              else{
                 String query="insert into Puesto(nom_p) values("'+nom.getText()+'");"; //Sino realiza la siguiente consulta
                 St.executeUpdate(query);
                 JOptionPane.showMessageDialog(null,"Registro agregado exitosamente");//Notifica que se agregó el registro exitosamente
              //cargardatosTabla("select*from puesto");
              cargardatosTabla("call TODOPue");//Manda a llamar al método cargar datos Tabla para actualizar
             //Pone los campos de texto en vacíos
              nom.setText("");
              busqueda.setText("");
              //Pone los botones de modificar y eliminar desactivados si están activados
               if(modificar.isEnabled() && eliminar.isEnabled()){
                  modificar.setEnabled(false);
                  eliminar.setEnabled(false);
               }
              }//
              //Cierra la conexión con la base de datos
              St.close();
              Rs.close();
              Con.close();
              autocompleta.removeAllItems();//Remueve todos los registros del autocompleta
              autocompletaDatos();
             } catch (Exception e) {//Si ocurre algún error se atrapa aquí
                 JOptionPane.showMessageDialog(null,e);
             }
         }//if de confirmacion
      }
    }
```

Botón Consultar Puestos



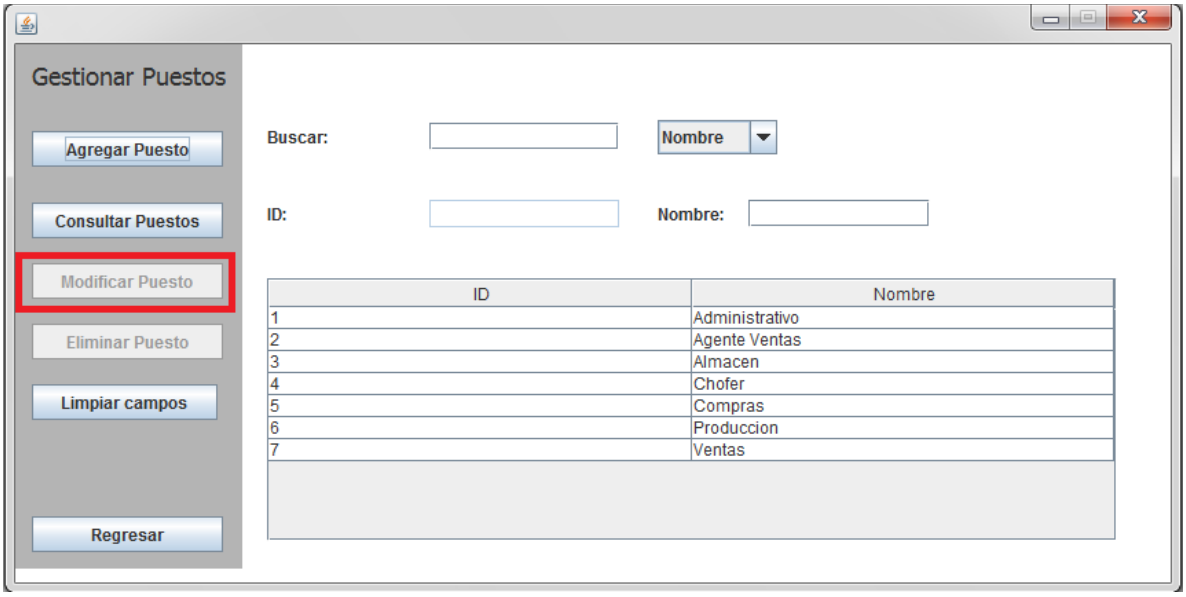
La función del botón Consultar Puestos es la mandan a llamar al método "consultarActionPerformed" cada vez que se da clic sobre él, este método realiza lo siguiente obtiene lo que se tiene escrito en el campo de búsqueda esto puede ser un nombre o un id y lo manda como parámetro de entrada al método de "llenacampos" en donde primero revisa si el parámetro de entrada está vacío de ser así muestra el mensaje de campo de campo de búsqueda vacío, de lo contrario establecerá conexión con la base de datos del sistema, comparara porque medio se está realizando la consulta ya sea por el nombre del puesto o por el ID en cualquier caso realiza una búsqueda en la base de datos con el parámetro de entrada sino lo encuentra entonces mostrara el mensaje de "No existe registro", en caso de que lo encuentre mandara llamar el método "cargardatos", cerrara la conexión a la base de datos y pondrá activos los botones de modificar y eliminar los cuales se encontraban inactivos.

Código:

```
private void consultarActionPerformed(java.awt.event.ActionEvent evt) {
      String parametro=busqueda.getText();//se obtiene el parámetro de entrada
      llenacampos(parametro);//Se envía el parámetro de entrada al método de llenacampos
    }

public void llenacampos(String parametro){
     //Revisa si el parámetro de entrada esta vacío
      if(parametro.isEmpty()){
        JOptionPane.showMessageDialog(null,"Campo de busqueda vacio"); //Si lo esta muestra el siguiente mensaje
      }
      else{
         try {
          //En caso contrario realiza la conexión a la base de datos
          Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
          St=Con.createStatement();
          String query="";
          //Declara una variable del tipo booleana y la inicia en true
```

```
        boolean existe;
        //Revisa de qué manera se está realizando la búsqueda seleccionada en el ComboBox
        if(ComboBusqueda.getSelectedItem()=="Nombre"){
          Rs = St.executeQuery("CALL BNOMPue('"+parametro+"');");// si se tiene el nombre en el ComboBox
        }
        else if(ComboBusqueda.getSelectedItem()=="ID"){
          parametro=quitaEspacios(parametro);//quitamos el espacio de más que se agregó al rellenar el autocompleter
          Rs = St.executeQuery("CALL BIDPue("+Integer.parseInt(parametro)+");"); // si se tiene el ID en el ComboBox
        }
            //a continuación nos damos cuenta si existe o no
             if(Rs.last()){existe=true;}
             else{
               JOptionPane.showMessageDialog(null,"No existe el registro");//Sino existe muestra el siguiente mensaje
               existe=false;//Pone la variable booleana en false
             }
            //revisa que la variable booleana este en true
           if(existe==true){
              if(ComboBusqueda.getSelectedItem()=="ID"){
                 query="CALL BIDPue("+Integer.parseInt(parametro)+");"; //De ser así realiza la siguiente consulta si esta
seleccionado ID
              }
              else if(ComboBusqueda.getSelectedItem()=="Nombre"){
                 query="CALL BNOMPue('"+parametro+"');"; //Si esta seleccionado el nombre esta
              }
              Rs=St.executeQuery(query);
              //Llama al método cargardatos
              cargardatos();
             //Activa los botones de modificar y eliminar
              modificar.setEnabled(true);
              eliminar.setEnabled(true);
              //Limpia el campo de búsqueda
              busqueda.setText("");
         }

         //Cierra la conexión al sistema
         St.close();
         Rs.close();
         Con.close();
         //Si ocurre alguna excepción esta se atrapa aquí
       } catch (Exception e) {
         JOptionPane.showMessageDialog(null,e);
       }
     }
   }
```

Botón Modificar Puesto



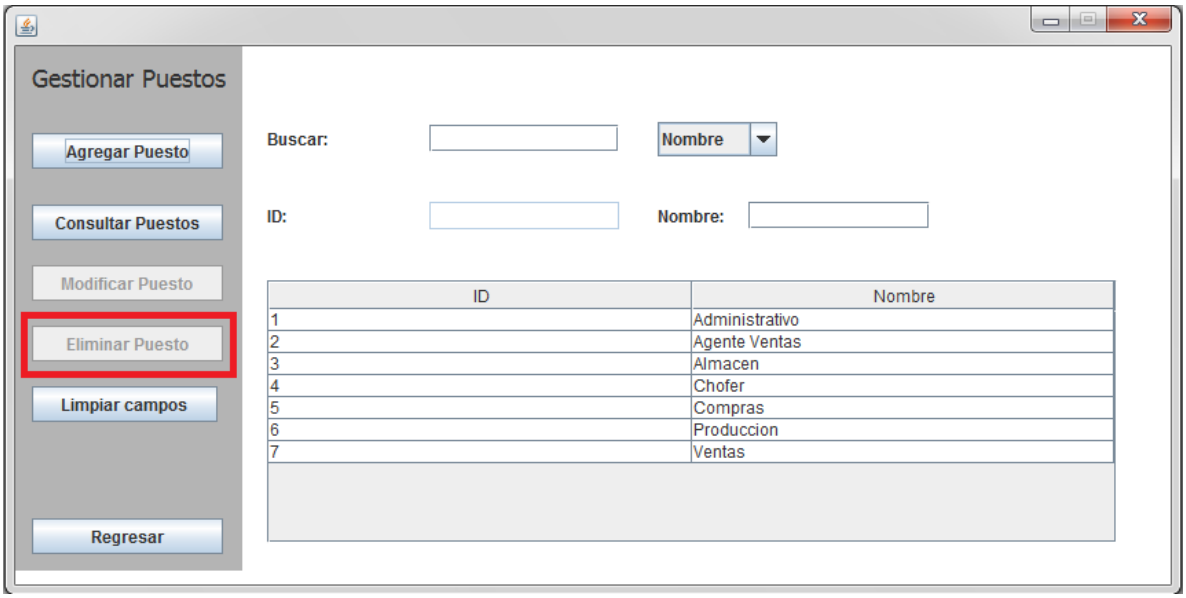
La función del botón Modificar Puesto es la de llamar al método "modificarActionPerformed" cada vez que se dé clic sobre ella mientras este activa, este método realiza lo siguiente revisa que si se tienen campos vacíos de ser así mostrara un mensaje de "Campos vacíos" de lo contrario mostrara un mensaje de "¿Estás seguro que deseas modificar este registro?" el cual es un mensaje de selección con las opciones si y no si se selecciona que si entonces establecerá una conexión con la base de datos y ejecuta un "update" al registro con la información proporcionada en la consulta previamente realizada, cierra la conexión con la base de datos y notifica al usuario que el registro se modificó exitosamente, limpia los campos de texto que se tienen en la interfaz, quita los componentes del autocompleter y los carga de nuevo ya actualizados, bloquea los botones de modificar y eliminar hasta que se realice otra búsqueda.

Código:

```
   private void modificarActionPerformed(java.awt.event.ActionEvent evt) {
//Se revisa si se tienen campos vacíos
      if(nom.getText().isEmpty() || id.getText().isEmpty()){JOptionPane.showMessageDialog(null,"Campos vacios");}
//Si se tienen muestra este mensaje
      else{
         Object [] opciones ={"Aceptar","Cancelar"};
//De lo contrario mostrara el siguiente mensaje de selección
         int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estas seguro de modificar este registro?","Mensaje de
Confirmacion",
         JOptionPane.YES_NO_OPTION,
         JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
//Si se selecciona si entonces realizara lo siguiente
         if (eleccion == JOptionPane.YES_OPTION)
         {
            try {
//Establece conexión con la base de datos
               Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
               St=Con.createStatement();
               String nombre=nom.getText();
```

```
//Ejecuta la siguiente consulta
                String query="CALL ModiPue("'+nombre+"',"'+Integer.parseInt(id.getText())+"');";
                //String query="update Puesto set nom_p="'+nom.getText()+"' where
ID_Pue="'+Integer.parseInt(id.getText())+"';";
                St.executeUpdate(query);
//Cierra conexión con la base de datos
                St.close();
                Con.close();
//Notifica al usuario que el registro se modificó exitosamente
                JOptionPane.showMessageDialog(null,"Registro modificado exitosamente");
                //cargardatosTabla("select*from Puesto");
//Llama al método cargardatosTabla
                cargardatosTabla("call TODOPue");
//Limpia los campos de búsqueda
                nom.setText("");
                id.setText("");
//Desactiva los botones de modificar y eliminar
                modificar.setEnabled(false);
                eliminar.setEnabled(false);
//Remueve los ítems del autocompleta
                autocompleta.removeAllItems();
//Los vuelve a cargar para actualizarlos
                autocompletaDatos();
//Si ocurre una excepción esta se atrapa aquí
            }   catch (Exception e) {
                JOptionPane.showMessageDialog(null,e);
                }
            }//if de confirmación
        }
    }
```

Botón Eliminar Puesto



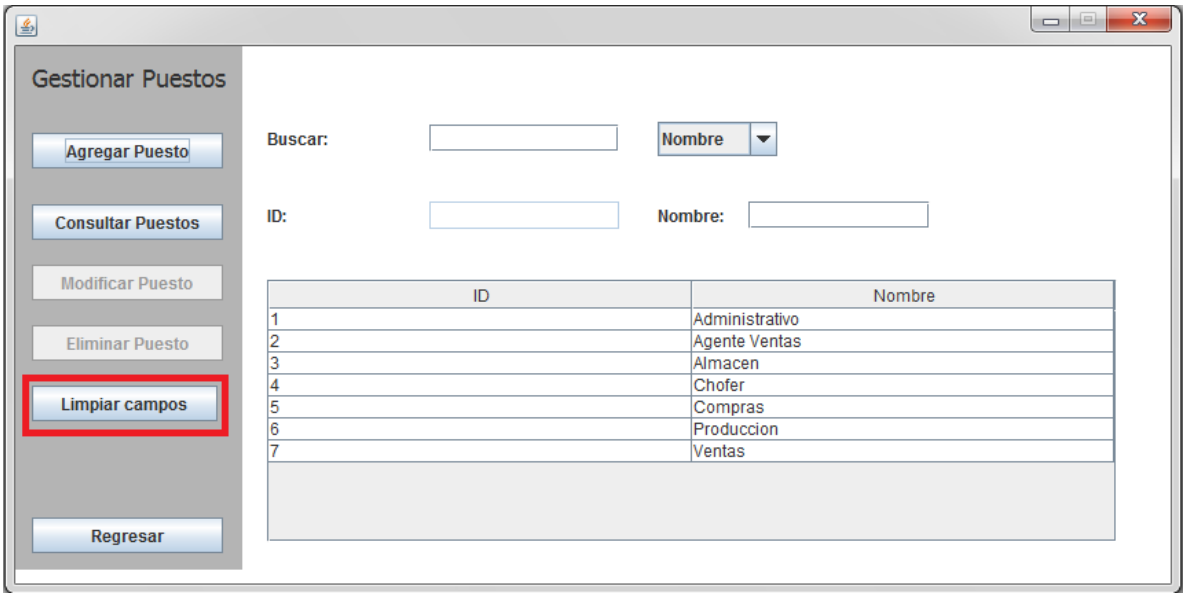
La función del botón Eliminar Puestos es la de mandar a llamar al método de "eliminarActionPerformed" cada vez que se dé clic sobre él, este método realiza lo siguiente revisa que si se tienen campos vacíos de ser así mostrara el mensaje de "Campos vacíos" de

lo contrario mostrara un mensaje de selección con las opciones de “Si” o “No” si se selecciona “Si” entonces se establecerá conexión con la base de datos, realizara una consulta la cual eliminara el registro encontrado en la búsqueda realizada anteriormente, cerrara conexión con la base de datos y mostrara un mensaje de “Registro eliminado exitosamente”, se limpiaran los campos de texto en la interfaz, se bloquearan los botones de modificar y eliminar y se actualizarán los registros que se tienen en el autocompleter de la siguiente manera, quitara los registros actuales y los volverá a cargar ya actualizados.

Código:

```
private void eliminarActionPerformed(java.awt.event.ActionEvent evt) {
//Revisa que no se tengan campos vacíos
     if(nom.getText().isEmpty()){JOptionPane.showMessageDialog(null,"Campos vacios");}//Si se tienen se mostrara el
siguiente mensaje
      else{
//De lo contrario se mostrara el siguiente mensaje de selección
         Object [] opciones ={"Aceptar","Cancelar"};
          int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estas seguro de eliminar este registro?","Mensaje de
Confirmacion",
          JOptionPane.YES_NO_OPTION,
          JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
          //Si se selecciona que si en dicho mensaje entonces realiza lo siguiente
          if (eleccion == JOptionPane.YES_OPTION)
          {
              try {
//Establecerá conexión con la base de datos
                  Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
                  St=Con.createStatement();
//Se ejecuta la siguiente consulta la cual elimina el registro introducido en la consulta
                  String query="CALL EliminarPue("'+Integer.parseInt(id.getText())+'");";
                  St.executeUpdate(query);
//Cierra conexión con la base de datos
                  St.close();
                  Con.close();
//Notifica al usuario que el registro se eliminó correctamente
                  JOptionPane.showMessageDialog(null,"Registro eliminado exitosamente");
                  //cargardatosTabla("select*from puesto");
 //Llama al método cargardatosTabla para actualizar los registros
                  cargardatosTabla("call TODOPue");
                  //Limpia los campos de texto de la interfaz
                  nom.setText("");
                  id.setText("");
//Desactiva los botones de modificar y eliminar hasta que se realice una nueva consulta
                  modificar.setEnabled(false);
                  eliminar.setEnabled(false);
//Remueve los ítems del autocompleter para actualizarlos
                  autocompleta.removeAllItems();
//Carga los datos en el autocompleter ya actualizados
                  autocompletaDatos();
//Si ocurre una excepción durante el proceso esta se atrapa aquí
              }   catch (Exception e) {
                  JOptionPane.showMessageDialog(null,e);
                  }
          }//if de confirmacion
     }
}
```

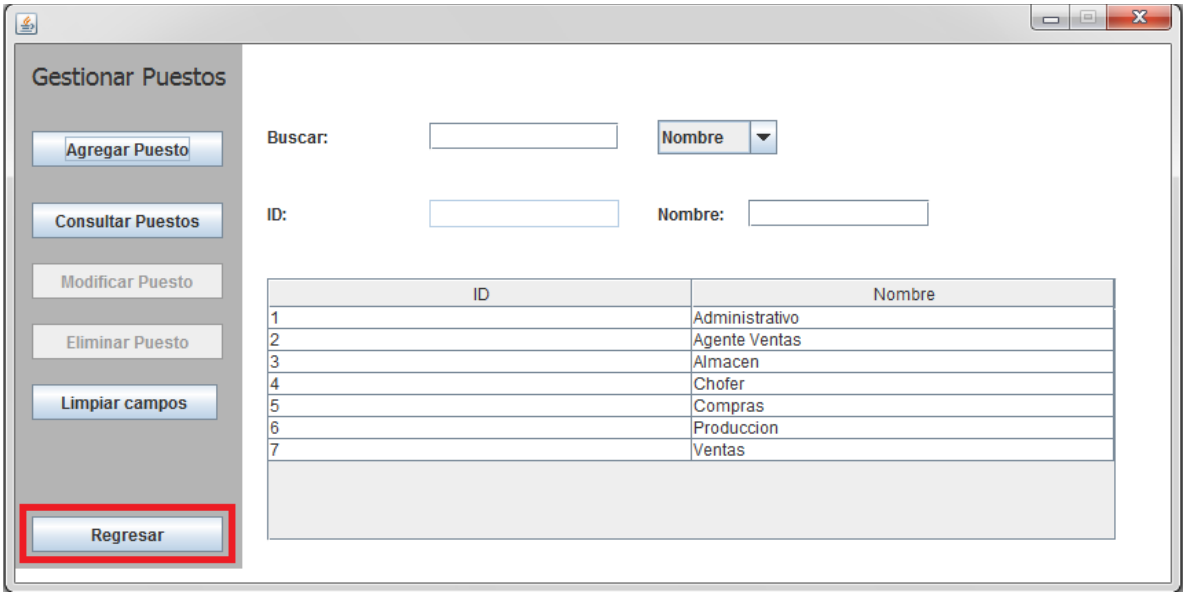
Botón Limpiar Campos



La función de Limpiar campos es de llamar al método "limpiarActionPerformed" cada vez que se da clic sobre él, este método realiza lo siguiente al campo de texto donde se introdujo los parámetros de la búsqueda se pone en vacío, de igual manera lo hace con los campos donde se muestran los datos del registro los cuales son nombre e id del puesto, por ultimo bloquea los botones de modificar y eliminar.

```
    //Método que se llama cada vez que se da clic sobre Limpiar Campos
        private void limpiarActionPerformed(java.awt.event.ActionEvent evt) {
   //Pon el campo de texto de búsqueda en vacío
   busqueda.setText("");
   //Pone el campo de texto de ID en vacío
   id.setText("");
   //Pone el campo de texto de nombre en vacío
   nom.setText("");
   //Bloquea los botones de modificar y eliminar
   modificar.setEnabled(false);
   eliminar.setEnabled(false);
   //cargardatosTabla("select*from puesto");
   //Llama al método cargardatosTabla
   cargardatosTabla("call TODOPue");
}
```

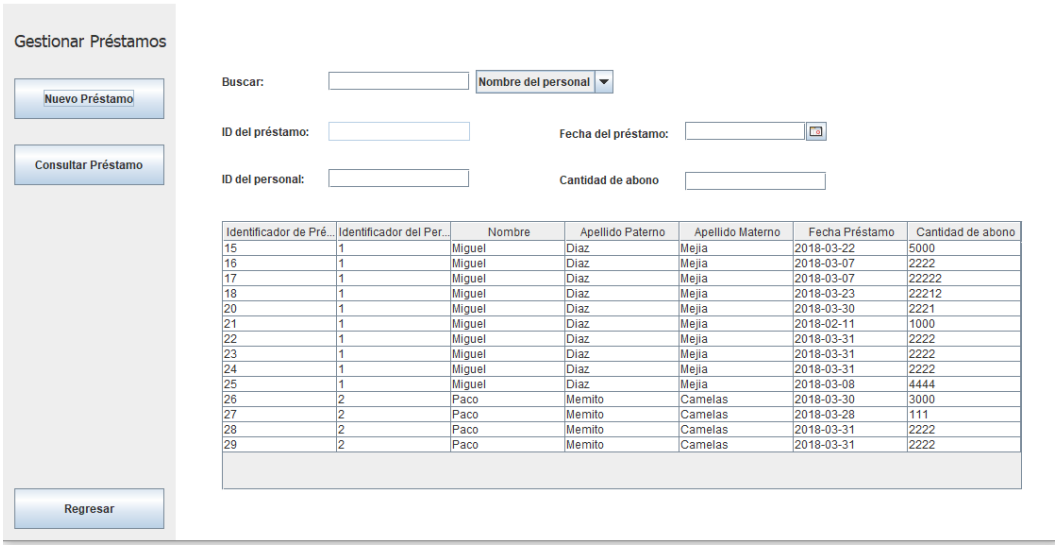
Botón Regresar Puestos



La función del botón regresar es llamar la método "jButton5ActionPerformed" cada vez que se dé clic sobre él, la función de este método es la de abrir la interfaz del menú principal del sistema y quitar la interfaz de Gestionar Puestos

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        //Se abre la interfaz de Menú principal
        new ControlPanel().setVisible(true);
        //Se liberan los recursos utilizados en esta interfaz y se cierra
        this.dispose();
    }
```

Módulo De Gestionar Préstamos.



Importaciones

Se deberán importar las siguientes librerías ya que son necesarias, las más importantes son la 2da, 3era, y 4ta ya que estas establecen la conexión con la base, y son utilizadas para consultas entre otras cosas.

```
package dsmafornuesau;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.Calendar;
import com.mxrck.autocompleter.AutoCompleterCallback;
import com.mxrck.autocompleter.TextAutoCompleter;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.ZoneId;
import java.util.Date;
```

Método constructor

El siguiente método sirve para crear el método constructor.

```
    public GestionarPrestamos() { //Se crea el método constructor
        initComponents();
        setLocationRelativeTo(null);
        getContentPane().setBackground(new java.awt.Color(255,255,255));  //Se define que el color de fondo sea blanco.
        setResizable(false);  //Se define que no se pueda cambiar el tamaño.
        //autocompleta= new TextAutoCompleter(busqueda);
        lista = new TextAutoCompleter(busqueda, new AutoCompleterCallback() {
            @Override
            public void callback(Object selectedItem) { //Define un método, para que cada vez que se seleccione con clic o
enter
```

```
            consultar((String)selectedItem);
            String sentenciaTabla="SELECT prestamos.ID_Pres, personal.ID_Per,
personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN
prestamos ON personal.ID_Per=prestamos.ID_Per and concat(personal.nom_p,' ',personal.ap,' ',personal.am) =
'"+(String)selectedItem+"';";  //Se realiza esta consulta.
            tabla(sentenciaTabla);
          }
        });
      ValidaFecha();  //Se invoca el método de ValidaFecha
   }   public GestionarPrestamos() { //Se crea el método constructor
      initComponents();
      setLocationRelativeTo(null);
      getContentPane().setBackground(new java.awt.Color(255,255,255));  //Se define que el color de fondo sea blanco.
      setResizable(false);  //Se define que no se pueda cambiar el tamaño.
      //autocompleta= new TextAutoCompleter(busqueda);
      lista = new TextAutoCompleter(busqueda, new AutoCompleterCallback() {
         @Override
         public void callback(Object selectedItem) { //Define un método, para que cada vez que se seleccione con clic o
enter
            consultar((String)selectedItem);
            String sentenciaTabla="SELECT prestamos.ID_Pres, personal.ID_Per,
personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN
prestamos ON personal.ID_Per=prestamos.ID_Per and concat(personal.nom_p,' ',personal.ap,' ',personal.am) =
'"+(String)selectedItem+"';";  //Se realiza esta consulta.
            tabla(sentenciaTabla);
          }
        });
      ValidaFecha();  //Se invoca el método de ValidaFecha
   }
```

Método GetFormatoFecha

Este método se manda a llamar cada vez que se ocupe obtener un formato de fecha correcto del calendario que está asignado en el módulo de préstamos.

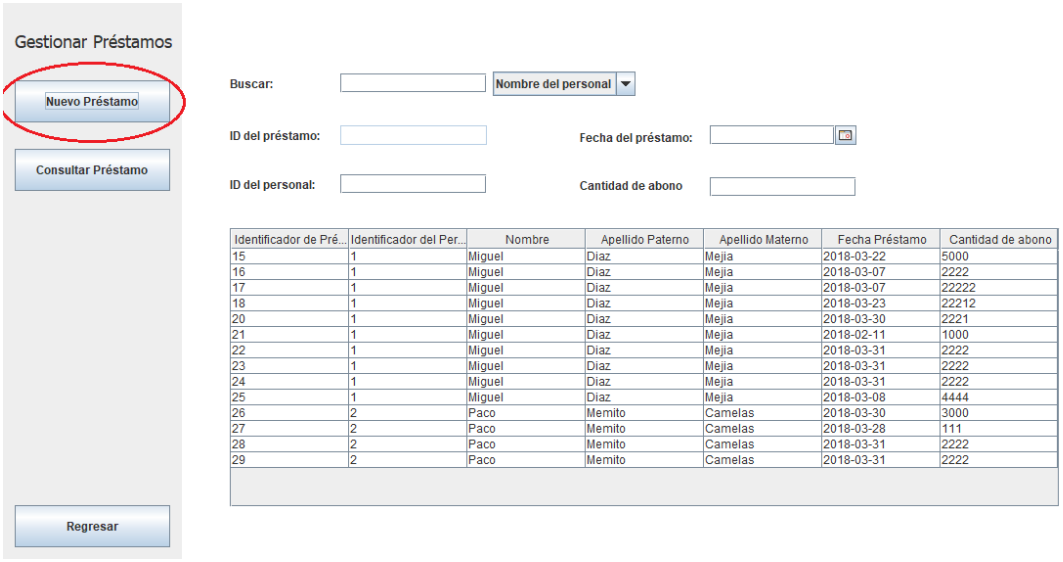
```
//método que obtiene el formato correcto de la fecha
    public String GetFormatoFecha(String nombDateChooser){
      //recibe un parámetro que indica de que calendario obtendrá la fecha
      String formato="";
      String dia,mes,año;
      if(nombDateChooser.equals("fecha")){
          //se obtiene individualmente el día, mes y año del calendario
          dia = Integer.toString(fecha.getCalendar().get(Calendar.DAY_OF_MONTH));
          mes = Integer.toString(fecha.getCalendar().get(Calendar.MONTH) + 1);
          año = Integer.toString(fecha.getCalendar().get(Calendar.YEAR));
          //se define el formato de fecha
          formato = (año + "-" + mes+ "-" + dia);
      }
      //se regresa el formato correcto de la fecha para su uso en otro lugar
      return formato;
   }
```

Método ValidaFecha

El siguiente método es utilizado para validar la fecha del préstamo, para que no se puedan ingresar préstamos a futuro o en el pasado.

```
    public void ValidaFecha()
    {
     Calendar cal= Calendar.getInstance();
     LocalDate ahora= LocalDate.now(); //Se toma la fecha del equipo
     Date date= Date.from(ahora.atStartOfDay(ZoneId.systemDefault()).toInstant());
     cal.add(Calendar.DAY_OF_YEAR,-0);
     Date max =cal.getTime();  // Se iguala la fecha máxima al día de hoy.
     fecha.setSelectableDateRange(max,date); //Se define que solo se puede seleccionar las fechas en un rango de entre las
variables max, y date
    }
```

Botón Nuevo Préstamo.



Método jButton3ActionPerformed

Cuando se presiona el botón de "Nuevo Préstamo", se inserta en la base de datos que se encuentran en los campos de la interfaz.

```
   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {    //Botón para nuevo préstamo
     String fechap;            //Se define la variable fechap
     String condicion="";  //Se define la variable condicion
    if(fecha.getDate()==null|| abono.getText().isEmpty()||idp.getText().isEmpty())  //Se compara lo que contiene el campo de fecha, abono,y idp, si alguno de estos esta vacío
     {
        JOptionPane.showMessageDialog(null,"Campos Vacios"); //Muestra el siguiente mensaje
     }
     else{
     Object [] opciones ={"Aceptar","Cancelar"}; //Se crea un arreglo con los objetos, aceptar y cancelar
     int eleccion = JOptionPane.showOptionDialog(rootPane,"Estas seguro de agregar un nuevo registro","Mensaje de Confirmacion",
     JOptionPane.YES_NO_OPTION,
     JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");//Se crea un mensaje de confirmación con las opciones aceptar, y cancelar
     if(eleccion == JOptionPane.YES_OPTION)  //Si la selección fue Afirmativa
     {
     try{
        String busid=idp.getText();   //Se crea y se iguala la variable busid a lo que contiene el campo idp
        int abonop=Integer.parseInt(abono.getText());  // Se crea y se iguala la variable abonop a lo que contiene el campo abono
        fechap=GetFormatoFecha("fecha");  //Se iguala la variable fechap a lo que retorna el metodo GetFormatoFecha

        Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza la conexión con la base de datos
        St=Con.createStatement();

        //se checa que el personal a recibir préstamo este activo
        Rs = St.executeQuery("select condicion as Condi from personal where ID_Per="+busid+";"); //Se realiza una consulta donde se selecciona la condición del personal donde su ID sea igual a la variable busid
        while(Rs.next()){condicion=Rs.getString("Condi");} //Se guarda el resultado

        if(condicion.equals("Activo")&&abonop>0){//si es activo y el abono es mayor a 0
```

```
            String query="insert into Prestamos(ID_Per,Fecha_Prestamo,cantidad)
values('"+busid+"','"+fechap+"','"+abonop+"');"; //Se inserta en la base de datos los valores de las variables busid, fechap,
y abonop
            St.executeUpdate(query);
            JOptionPane.showMessageDialog(null,"Registro agregado exitosamente");  //Se muestra el mensaje

             cargardatosTabla("SELECT prestamos.ID_Pres, personal.ID_Per,
personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN
prestamos ON personal.ID_Per=prestamos.ID_Per"); //Se realiza una búsqueda con los datos del personal y del préstamo
             idp.setText(""); //Se iguala la variable a vacío
             id.setText(""); //Se iguala la variable a vacío
             fecha.setCalendar(null); //Se iguala el calendario a nulo
             abono.setText(""); // Se iguala la variable a vacío
             busqueda.setText(""); //Se iguala la variable a vacío
          }else{//no es activo
           if(condicion.equals("Inactivo"))JOptionPane.showMessageDialog(null,"No se puede registrar un prestamo a un
trabajador dado de baja"); //Se muestra el mensaje
           else if(abonop<=0)JOptionPane.showMessageDialog(null,"La cantidad de abono debe ser mayor a cero"); // Si el e
abono es negativo o es igual a 0 se muestra el mensaje
           else JOptionPane.showMessageDialog(null,"El ID de personal no existe"); //Si no se cumple nada el personal no
existe
          }
          St.close();
          Rs.close();
          Con.close();  //Se cierra conexión con la base de datos
          autocompletaDatos();  // Se invoca el método autocompletaDatos
         } catch (Exception e){JOptionPane.showMessageDialog(null,e);} // Si ocurre algún error muestra el mensaje.
        }//if
       }//if
      }
```

Método cargardatosTabla

El siguiente método es utilizado para cargar los datos de la consulta, que se tienen en la base de datos dentro de la tabla.

```
public void cargardatosTabla(String consulta){  //Método para cargar los datos de la tabla
          try {
             DefaultTableModel tabla=new DefaultTableModel(); // Se crea un objeto de la clase tabla.
             tabla.setColumnIdentifiers(new Object[]{"Identificador de Préstamo","Identificador del
Personal","Nombre","Apellido Paterno","Apellido Materno","Fecha Préstamo","Cantidad de abono"});  //Se ingresan en
la tabla los identificadores
             Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza conexión a la base
de datos.
             St=Con.createStatement();
             Rs=St.executeQuery(consulta);
             int i=0; // Se crea la variable i
             while(Rs.next()){ //Mientras se tengan resultados
             tabla.addRow(new
Object[]{Rs.getString(1),Rs.getString(2),Rs.getString(3),Rs.getString(4),Rs.getString(5),Rs.getString(6),Rs.getString(7)})
; //Se pone el resultado dentro de una columna de la tabla
                          }
             jTable1.setModel(tabla); //Se invoca la jTable1 con la tabla
             Con.close();  // Se cierra conexión con la base de datos.
             St.close();
          }
             catch (Exception e) { // Si ocurre un error
             JOptionPane.showMessageDialog(null,e); // Muestra el mensaje.
         }
      }
```

Método tabla

El siguiente método es utilizado para cargar los datos en la tabla cuando se selecciona la opción de “Nombre del personal”, para realizar una consulta.

```
public void tabla(String consulta){  // Se recibe la consulta
  if(jComboBox1.getSelectedItem()=="Nombre del personal")cargardatosTabla(consulta); // si la selección del combobox fue "Nombre del personal", se invoca el método cargardatosTabla con la consulta
}
```

Método autocompletaDatos

El siguiente método es utilizado para añadir a una lista los datos que se quieren auto complementar.

```
public void autocompletaDatos(){      //Método autocompleta los datos
  lista.removeAllItems(); // Se remueven todos los items que se tenían en lista.
   try {
      Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza la conexión a la base de datos
      St=Con.createStatement();

      if(jComboBox1.getSelectedItem()=="ID del prestamo")Rs=St.executeQuery("select ID_Pres as parametro from prestamos;");   //Si se selecciono "ID del préstamo", se realiza una consulta que seleccione solo el ID del préstamo
      else if(jComboBox1.getSelectedItem()=="Nombre del personal")Rs=St.executeQuery("SELECT distinct concat(personal.nom_p,' ',personal.ap,' ',personal.am) as parametro FROM personal INNER JOIN prestamos ON personal.ID_Per=prestamos.ID_Per");  //Si la selección fue "Nombre del personal", se realiza una consulta que seleccione nombre, apellido paterno. y materno todo concatenado.

      while(Rs.next()){ //Mientras que se tenga resultado
        if(jComboBox1.getSelectedItem()=="ID del prestamo"){  // Si se seleccionó la opción de "ID del préstamo´"
          lista.addItem(Rs.getString("parametro")+" ");  // Añade a lista, la cadena parámetro más el resultado
        }
        else if(jComboBox1.getSelectedItem()=="Nombre del personal"){ // Si se sele la opción de "Nombre del personal"
          lista.addItem(Rs.getString("parametro"));    //Añade a la lista, la cadena parámetro
        }
      }
      St.close();
      Rs.close();
      Con.close();  // Se cierra  la conexión a la base de datos
    } catch (Exception e) { // Si se tiene algun error
      JOptionPane.showMessageDialog(null,e); // Muestra cual es el error.
    }
}
```

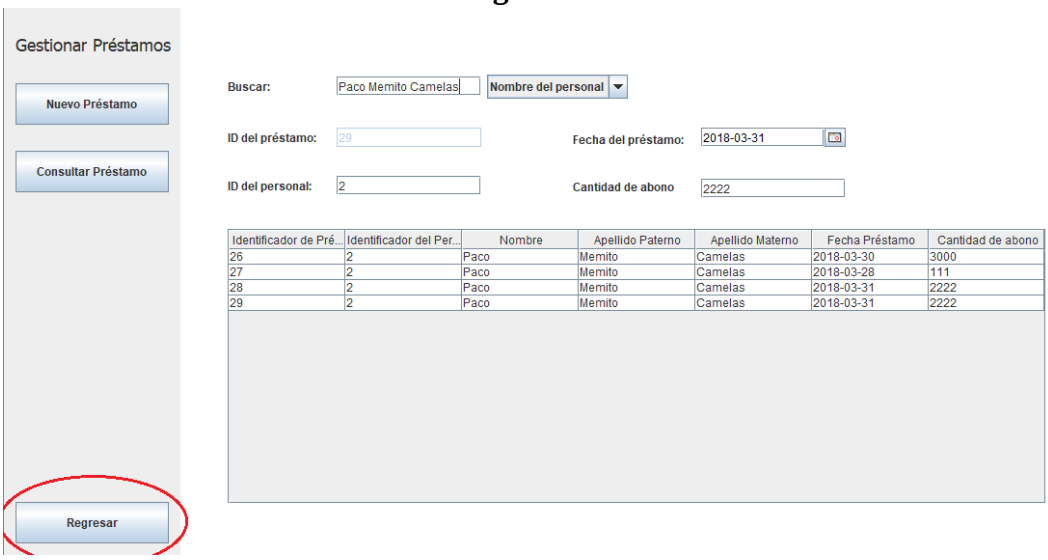
Método cargardatos

El siguiente método es utilizado para cargar los datos de la consulta realizada, dentro de los campos correspondientes.

```
public void cargardatos(){ // Método para cargar los datos dentro de los campos correspondientes
    try {
      while(Rs.next()){ // Mientras  se tenga resultado
      id.setText(Rs.getString(1)); // Manda al campo de id el resultado numero 1
      idp.setText(Rs.getString(2));// Manda al campo de idp el resultado numero 2
      Rs.getString(3); //Obtiene el resultado numero 3
      Rs.getString(4); //Obtiene  el resultado numero 4
      Rs.getString(5); //Obtiene el resultado numero 5
```

```
        Date date = new SimpleDateFormat("yyyy-MM-dd").parse(Rs.getString(6));fecha.setDate(date); // Se crea un objeto tipo fecha con su formato simple, y se pasa como el resultado numero 6
        abono.setText(Rs.getString(7));} // Se manda al campo abono el resultado numero 7
    } catch (Exception e) { //Si hay un error
        JOptionPane.showMessageDialog(null,"Error"+e);  //Muestra mensaje de error y que error fue.
    }
}
```

Botón Regresar Préstamo



Método jButton6ActionPerformed

Cuando se presiona el botón “Regresar”, se cierra la ventana actual y se cambia a visible, la interface de Panel de control.

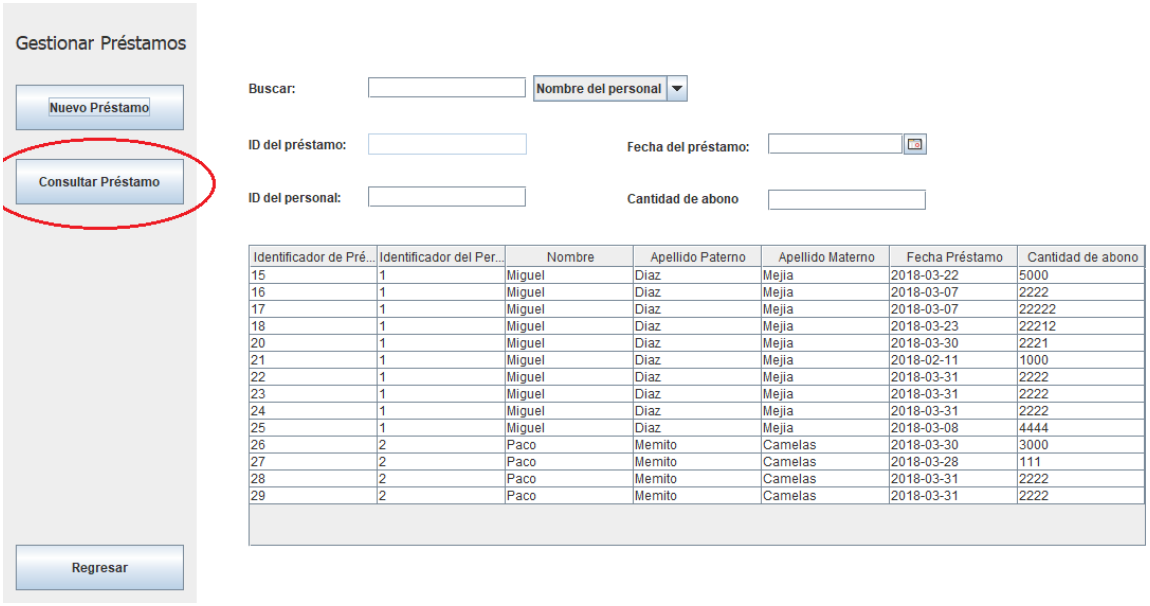
```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {      //Botón para regresar al menú principal
    new ControlPanel().setVisible(true);  // Se pone como visible la interface de ControlPanel
    this.dispose();
}
```

Método formWindowOpened

El siguiente método es utilizado para cargar los datos de la tabla cuando se ingresa al módulo de Gestionar Préstamos.

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {   // Método utilizado cuando se abre la ventana
    cargardatosTabla("SELECT prestamos.ID_Pres, personal.ID_Per, personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN prestamos WHERE personal.ID_Per = prestamos.ID_Per"); //Se invoca el método cargardatosTabla con los parámetros
    autocompletaDatos(); // Se invoca el método autocompletaDatos
}
```

Botón Consultar Préstamo



Método jButton4ActionPerformed

Al presionar el botón Consultar Préstamo, se realiza una consulta en la base de datos y se muestran los resultados en los campos correspondientes.

```
  private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {    //Botón consultar
     String parametro=busqueda.getText(); //Se crea la variable parámetro y se iguala a lo que contiene el campo de
buqueda.
     consultar(parametro); // Se invoca el método consultar mandado parámetro
  }
```

Método consultar

El siguiente método realiza una consulta, dependiendo de cual haya sido la selección del combobox, se realizará distintas formas, pero en ambas se muestra la misma información.

```
public void consultar(String parametro){   //Este método realiza la consulta
    if(parametro.isEmpty()) // Si parámetro esta vacío
      {JOptionPane.showMessageDialog(null,"Campo de busqueda vacio");} // Muestra mensaje
      else{ // Sino
       try{
          int idb=0; // Se crea la variable entera idb
          Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); // Se realiza conexión con la base
de datos
          St=Con.createStatement();
          String query;
          boolean existe;    // Se crea la variable booleana existe
          if(jComboBox1.getSelectedItem()=="Nombre del personal"){   //Si la selección del combobox fue "Nombre de
personal"
           Rs = St.executeQuery("SELECT DISTINCT concat(personal.nom_p,' ',personal.ap,' ',personal.am)  FROM
personal INNER JOIN prestamos WHERE personal.ID_Per = prestamos.ID_Per and concat(personal.nom_p,'
',personal.ap,' ',personal.am)= '"+parametro+"';");  //Se realiza la consulta teniendo como condición que se igual al
parámetro
          }
          else if(jComboBox1.getSelectedItem()=="ID del prestamo"){ //Si la selección del combobox fue "ID del
préstamo"
```

```
            parametro=quitaEspacios(parametro); //Se iguala la variable parámetro a la invocación del método quitaEspacios mandando parámetro
            Rs = St.executeQuery("SELECT prestamos.ID_Pres FROM personal INNER JOIN prestamos WHERE personal.ID_Per = prestamos.ID_Per and prestamos.ID_Pres="+Integer.parseInt(parametro)+";");   //Se realiza una consulta teniendo como condición que sea igual a la conversión de parámetro a entero.
          }
              //a continuación nos damos cuenta si existe o no
              if(Rs.last()){existe=true;}
              else{
                JOptionPane.showMessageDialog(null,"No existe el registro"); // Si no se cumple condición anterior se muestra el mensaje.
                existe=false; //Se iguala la variable existe a false
              }
             if(existe==true){   //Compara el valor de variable existe con true

              if(jComboBox1.getSelectedItem()=="ID del prestamo"){  //Si la selección del combobox fue "ID del préstamo"
                query="SELECT prestamos.ID_Pres, personal.ID_Per, personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN prestamos WHERE personal.ID_Per = prestamos.ID_Per and prestamos.ID_Pres="+Integer.parseInt(parametro)+";"; ////Se realiza una consulta teniendo como condición que sea igual a la conversión de parámetro a entero.
                Rs=St.executeQuery(query);
              }
              else if(jComboBox1.getSelectedItem()=="Nombre del personal"){     //Si la selección del combobox fue "Nombre del personal"
                query="SELECT prestamos.ID_Pres, personal.ID_Per, personal.nom_p,personal.ap,personal.am,prestamos.Fecha_Prestamo,prestamos.cantidad FROM personal INNER JOIN prestamos WHERE personal.ID_Per = prestamos.ID_Per and concat(personal.nom_p,' ',personal.ap,' ',personal.am)= '"+parametro+"'"; //Se realiza la consulta teniendo como condición que se igual al parámetro
                Rs=St.executeQuery(query);
              }
                cargardatos(); // Se invoca el método cargardatos
                busqueda.setText("");  //Se vacía el campo búsqueda
             }
           St.close();
           Rs.close();
           Con.close();   //Se cierra la conexión a la base de datos
         } catch (Exception e){JOptionPane.showMessageDialog(null,e);}    //Si ocurre un error, muestra error, y cual error fue.
       }
    }
```

Método quitaEspacios

El siguiente método es utilizado para quitar espacios de una cadena de caracteres.

```
public String quitaEspacios(String texto) { //Este metodo elimina espacios dentro de una cadena de caracteres
    java.util.StringTokenizer tokens = new java.util.StringTokenizer(texto); // Se crea un objeto de tipo Tokenizer
    StringBuilder buff = new StringBuilder();  //Se crea un objeto de tipo Builder
    while (tokens.hasMoreTokens()) { // Mientras se tengan tokens
       buff.append(" ").append(tokens.nextToken());
    }
    return buff.toString().trim(); // Se retorna lo que se tiene en el buffer
}
```

Método jComboBox1ActionPerformed

El siguiente método es utilizado para obtener la selección del usuario para realizar la consulta del préstamo, esta puede ser por nombre del personal, o por ID del préstamo.

```
   private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {    //Método para obtener la selección del usuario del ComboBox
       x=jComboBox1.getSelectedIndex();//Se iguala la variable x a la selección
       autocompletaDatos(); //Se invoca el método autocompletaDatos
       busqueda.setText("");//Se vacía el campo de búsqueda
   }
```

Método busquedaKeyTyped

El siguiente método es utilizado para mandar un sonido de error y eliminar el carácter ingresado cuando este no es válido.

```
private void busquedaKeyTyped(java.awt.event.KeyEvent evt) {
      if(jComboBox1.getSelectedItem()=="Nombre del personal"){ // Si la selección del combobox fue "Nombre del personal"
        String nombre=busqueda.getText(); //Se crea y se iguala la variable nombre al contenido del campo búsqueda
         char k=evt.getKeyChar(); // Se crea y se iguala la variable k a la tecla presionada
        if(Character.isLetter(k)||k=='.'||k==' '){ // Si la tecla presionada es punto o espacio
         if(nombre.length()>0){          // Si el tamaño de nombre es mayor a 0
           if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){ //Si el ultimo carácter menos uno del nombre es punto y espacio
            getToolkit().beep();//Invoca sonido de error de Windows
            evt.consume();//Elimina el carácter ingresado
           }
          else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){ //Si el ultimo carácter menos uno del nombre es punto y espacio
            getToolkit().beep();//Invoca sonido de error de Windows
            evt.consume();//Elimina el carácter ingresado
           }
           else if(nombre.length()==25){ //Si el largo del campo nombre es igual a 25
             getToolkit().beep();//Invoca sonido de error de Windows
             evt.consume();//Elimina el carácter ingresado
           }
         }
         else if(nombre.isEmpty()){ // Si el campo nombre está vacío
           if(k==' '||k=='.'){ // Si la variable k es igual a punto o espacio
             getToolkit().beep();//Invoca sonido de error de Windows
             evt.consume();  //Elimina el carácter ingresado
           }
         }
        }else{ //Sino
            evt.consume();
        }
        if(nombre.contains(".")&&k=='.'){ // Si el nombre contiene punto y
            getToolkit().beep();//Invoca sonido de error de Windows
            evt.consume();//Elimina el carácter ingresado
         }
      }
      else if(jComboBox1.getSelectedItem()=="ID del prestamo"){ //Si la selección del combobox fue "ID del préstamo"
        char k=evt.getKeyChar();
        String ID=busqueda.getText();     //Se crea y se iguala al contenido del campo búsqueda
        if(Character.isDigit(k)){ // Si carácter es igual a la variable k
          if(ID.length()>0){          //Si el tamaño de ID es mayor a 0
            if(ID.length()==10){ //Si el tamaño de ID es igual a 10
              evt.consume();//Elimina el carácter ingresado
             }
            }
         }
        else{
          evt.consume();//Elimina el carácter ingresado
        }//else
      }
```

```
}
```

Método idpKeyTyped

El siguiente método es utilizado para mandar un sonido de error y eliminar el carácter ingresado cuando este no es válido.

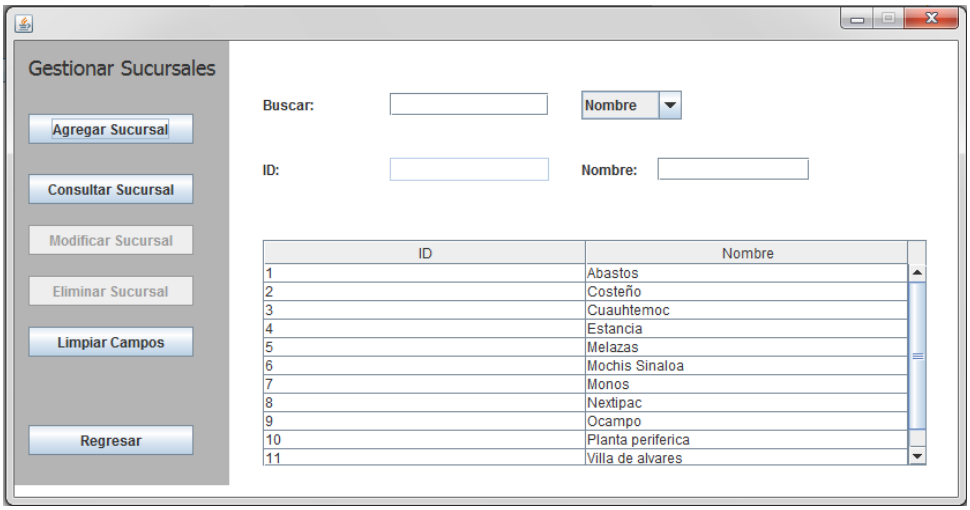
```
private void idpKeyTyped(java.awt.event.KeyEvent evt) {
    char k=evt.getKeyChar(); //Se iguala la variable k al carácter presionado
    String ID=idp.getText();    // Se crea y se iguala la variable ID al contenido del campo idp
    if(Character.isDigit(k)){ // Si el carácter ingresado es digito
      if(ID.length()>0){    //Si el tamaño de ID es mayor a 0
        if(ID.length()==10){ // Si el tamaño de ID es igual a 10
         evt.consume(); //Elimina el carácter ingresado
        }
       }
     }
    else{
      evt.consume(); //Elimina el carácter ingresado
    }//else
}
```

Método abonoKeyTyped

El siguiente método es utilizado para mandar un sonido de error y eliminar el carácter ingresado cuando este no es válido.

```
private void abonoKeyTyped(java.awt.event.KeyEvent evt) {
    char k=evt.getKeyChar();   //se crea la variable k y se iguala a la tecla presionada
    String ABONO=abono.getText();        // Se crea la variable abono y se iguala a lo que contiene el campo abono
    if(Character.isDigit(k)){ // Si k es igual a digito
     if(ABONO.length()>0){         //Si abono
       if(ABONO.length()==5){ //Si el tamaño de abono es igual a 5
        evt.consume(); //Elimina el carácter ingresado
       }
     }
    }
    else{
      evt.consume();//Elimina el carácter ingresado
    }
}
```

Modulo Gestionar Sucursales



Este módulo utiliza las siguientes librerías todas son requeridas para el correcto funcionamiento de todas las funciones:

```
//Importación de las librerías requeridas para las funciones del sistema
import com.mxrck.autocompleter.AutoCompleterCallback;
import com.mxrck.autocompleter.TextAutoCompleter;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Este es el método constructor de la clase Gestionar Sucursales:

```
//Este es el método constructor del módulo Gestionar Sucursales
   public Sucursales() {
      //Método para que los componentes del frame
      initComponents();
      //Para evitar que los componentes se muevan de lugar por x razón
      setLocationRelativeTo(null);
      //Se establece el color de fondo del panel
      getContentPane().setBackground(new java.awt.Color(255,255,255));
      //Se desactiva que se pueda modificar el tamaño de la ventana
      setResizable(false);
     //Se crea un auto completer
      autocompleta = new TextAutoCompleter(busqueda, new AutoCompleterCallback() {
      //Se crea el método para obtener los ítems seleccionados y utilizarlos para las consultas
      public void callback(Object selectedItem) {
      //Se llama al método consultar con el parámetro del ítem seleccionado
            consultar((String)selectedItem);
        }
       });
   }
```

Este módulo también hace uso de las siguientes variables globales que son utilizadas por la mayoría de funciones que tiene:

```
//Declaración de las variables globales que se utilizaran por la mayoría de las funciones

  TextAutoCompleter autocompleta;
   Connection Con;
   Statement St;
   ResultSet Rs;
   static int x=0;
```

El modulo utiliza los siguientes métodos que son utilizados por la mayoría de funciones para su correcto funcionamiento.

Método quitaEspacios

```
//Este método elimina los espacios de más de una cadena mandada como parámetro de entrada.
public String quitaEspacios(String texto) {
       java.util.StringTokenizer tokens = new java.util.StringTokenizer(texto);
       StringBuilder buff = new StringBuilder();
       while (tokens.hasMoreTokens()) {
          buff.append(" ").append(tokens.nextToken());
       }
       return buff.toString().trim();
```

```
    }
```

Método formWindowOpened

```
//Este método se carga de manera automática cada vez que se abre la interfaz de Gestionar Sucursales
 private void formWindowOpened(java.awt.event.WindowEvent evt) {
     //Llama al método cargardatosTabla con la consulta de cargar todo de la tabla Sucursal
     cargardatosTabla("select * from Sucursal");
    //Llama al método de autocompletaDatos para cargar los datos en el autocompleter.
     autocompletaDatos();
}
```

Método combobusquedaActionPerformed

```
       //Este método se llama cada vez que se da clic sobre el ComboBox destinado para la búsqueda
 private void combobusquedaActionPerformed(java.awt.event.ActionEvent evt) {
       //Se declara una variable con el valor decimal que tiene el ComboBox
       x=combobusqueda.getSelectedIndex();
       //Revisa que se tenga seleccionado algo en el ComboBox
       if(x==0 || x==1)
       {
       //De ser así entonces pondrá el campo de texto de búsqueda en vacío
          busqueda.setText("");
       //Y llamara al método cargardatosTabla con la consulta de obtener todo los datos y registros de Sucursal
          cargardatosTabla("select * from Sucursal");
       }
       //Remueve todos los ítem se autocompleta
       autocompleta.removeAllItems();
       //Los vuelve a cargar llamando al método autocompletaDatos
       autocompletaDatos();
   }
```

Método nomKeyTyped

```
      //Se manda a llamar a este método automáticamente para validar que sea un carácter correcto
             private void nomKeyTyped(java.awt.event.KeyEvent evt) {
       //primero se guarda en la cadena nombre lo que hay en el campo
       //de texto antes del nuevo carácter ingresado
       String nombre=nom.getText();
       //se obtiene solo el carácter ingresado
        char k=evt.getKeyChar();
        //solo si el carácter ingresado es una letra, un punto o un espacio se
        //dejara ingresar y se analizara el orden correcto para la validación
        if(Character.isLetter(k)||k=='.'||k==' '){
        //si el campo de texto no está vacío
         if(nombre.length()>0){
         //si el último carácter en el campo de texto es un punto
         //y el carácter ingresado es diferente de un espacio
         //entonces no se deja ingresar este carácter
           if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
           getToolkit().beep();
           evt.consume();
          }
           //si el último carácter en el campo de texto es un espacio
          //y el carácter ingresado es un espacio o un punto
          //entonces no se deja ingresar este carácter
          else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
           getToolkit().beep();
           evt.consume();
          }
          //no se deja ingresar un carácter si en el campo de texto ya hay 22 caracteres
           else if(nombre.length()==22){
             getToolkit().beep();
             evt.consume();
           }
```

```
        }
        //si el campo de texto si está vacío no se puede
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
            getToolkit().beep();
            evt.consume();
          }
        }
       }else{
            //si el carácter es algo diferente de una letra, punto o espacio
            //no se deja ingresar el carácter
            //getToolkit().beep();
            evt.consume();
       }
            //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
            getToolkit().beep();
            evt.consume();
        }
 }
```

Método busquedaKeyTyped

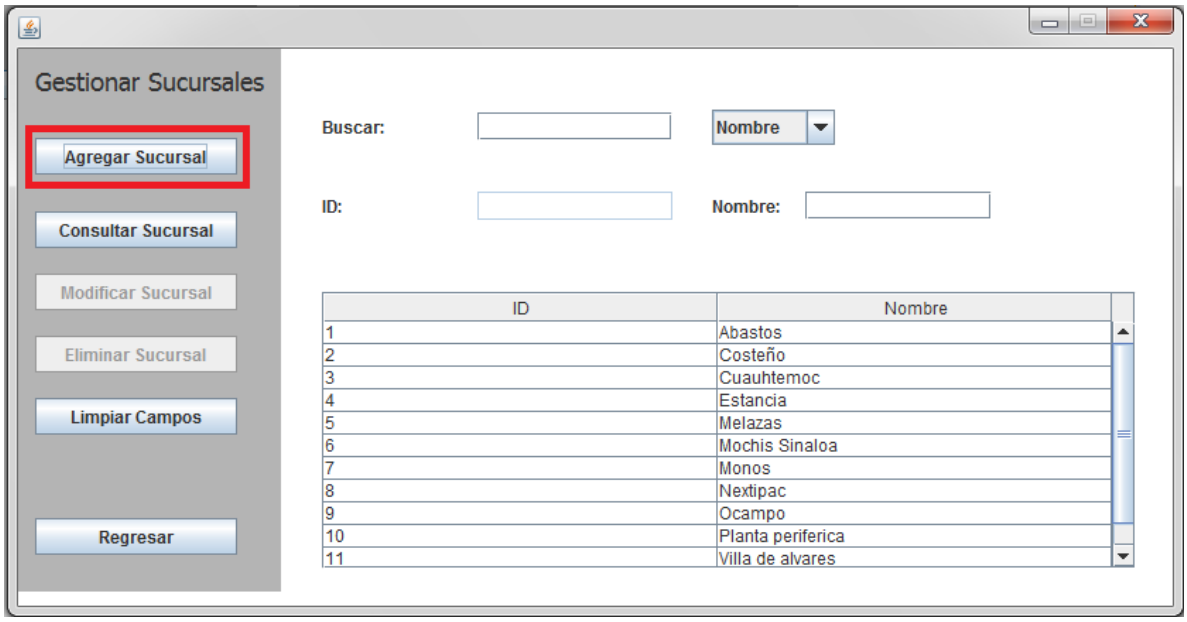
```
//se manda a llamar a este método automáticamente para validar que sea un carácter correcto
 private void busquedaKeyTyped(java.awt.event.KeyEvent evt) {
       //Se revisa si se tiene seleccionado nombre en el ComboBox
       if(combobusqueda.getSelectedItem()=="Nombre"){
      //primero se guarda en la cadena nombre lo que hay en el campo de texto antes del nuevo carácter ingresado
        String nombre=busqueda.getText();
      //se obtiene solo el carácter ingresado
        char k=evt.getKeyChar();
       //solo si el carácter ingresado es una letra, un punto o un espacio se
       //dejara ingresar y se analizara el orden correcto para la validación
       if(Character.isLetter(k)||k=='.'||k==' '){
       //Si el campo de texto no está vacío
        if(nombre.length()>0){
          //si el último carácter en el campo de texto es un punto
          //y el carácter ingresado es diferente de un espacio
          //entonces no se deja ingresar este carácter
          if(nombre.charAt(nombre.length()-1)=='.'&&k!=' '){
          getToolkit().beep();
          evt.consume();
          }
          //si el último carácter en el campo de texto es un espacio
          //y el carácter ingresado es un espacio o un punto
          //entonces no se deja ingresar este carácter
          else if(nombre.charAt(nombre.length()-1)==' '&&(k==' '||k=='.')){
           getToolkit().beep();
           evt.consume();
          }
          //no se deja ingresar un carácter si en el campo de texto ya hay 22 caracteres
           else if(nombre.length()==22){
             getToolkit().beep();
             evt.consume();
           }
        }
        //si el campo de texto si está vacío no se puede
        //ingresar espacios o puntos
        else if(nombre.isEmpty()){
          if(k==' '||k=='.'){
            getToolkit().beep();
            evt.consume();
          }
```

```
        }
       }else{
          //si el carácter es algo diferente de una letra, punto o espacio
          //no se deja ingresar el carácter
           //getToolkit().beep();
           evt.consume();
       }
       //si el campo de texto ya contiene un punto ya no deja ingresar otro
       if(nombre.contains(".")&&k=='.'){
           getToolkit().beep();
           evt.consume();
        }
    }
     //Se comprueba si esta seleccionado el ID en el ComboBox
    else if(combobusqueda.getSelectedItem()=="ID"){
    //Se obtiene solo el carácter ingresado
      char k=evt.getKeyChar();
      //primero se guarda en la cadena nombre lo que hay en el campo
      //de texto antes del nuevo carácter ingresado
      String ID=busqueda.getText();
      //Se revisa si la cadena está compuesta de dígitos
      if(Character.isDigit(k)){
      //Se comprueba si está vacía la cadena
         if(ID.length()>0){
      //De igual manera que no pase de una longitud de 10 dígitos
           if(ID.length()==10){
      //Si el carácter es algo diferente de una letra, punto o espacio
           //no se deja ingresar el carácter
           //getToolkit().beep();
           evt.consume();
           }
         }
      }
      else{
       //Si el carácter es algo diferente de una letra, punto o espacio
        //no se deja ingresar el carácter
       evt.consume();
      }//else
    }
  }
```

Método cargardatos

```
    //Este método carga los datos a los campos de texto correspondientes
            public void cargardatos(){

     try {
    //Mientras encuentre otro dato del registro
        while(Rs.next()){
    //Obtiene el ID del puesto y lo pone en el campo de texto del ID
        id.setText(Rs.getString(1));
    //Obtiene el Nombre del puesto y lo pone en el campo de Texto
        nom.setText(Rs.getString(2));}
    //Si ocurre alguna excepción esta se atrapa aquí
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Error"+e);
    }

  }
```

Botón Agregar Sucursal



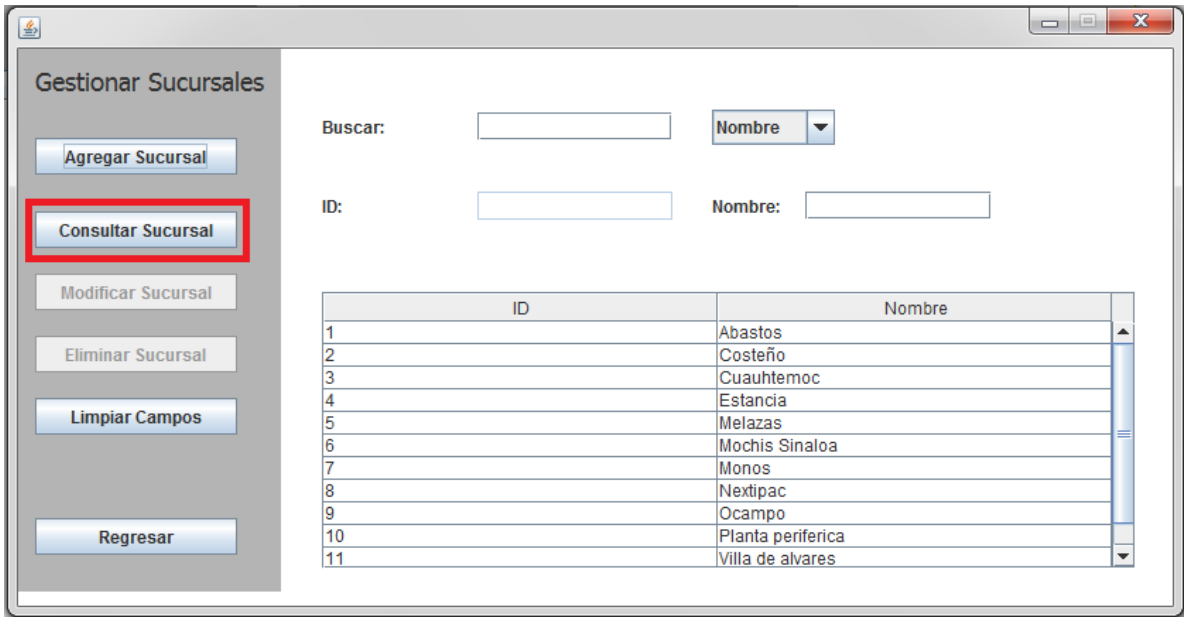
La función del botón agregar sucursal es la de llamar al método "jButton1ActionPerformed" cada vez que se da clic sobre él, dicho método agrega una sucursal al sistema de la siguiente manera revisa que el campo de texto de nombre no este vacío, si lo está entonces muestra un mensaje diciendo "Campos vacíos", de lo contrario se creara un objeto al cual se le llenan las opciones las cuales se mostraran en un mensaje de selección, se mostrara el mensaje de selección de "Estas seguro que de agregar un nuevo registro" con las opciones de Si y No, si el usuario selecciona Si entonces tomara lo que se tiene en el campo de nombre, establecerá conexión con la base de datos del sistema y realizara una consulta para comprobar si existe un registro con ese nombre, si existe entonces mostrara un mansaje de "Ya existe un registro con ese nombre", de lo contrario realizara una consulta agregando el registro con el nombre ingresado, se limpiaran los campos de la interfaz, se cerrara la conexión con la base de datos del sistema y se bloquearan los botones de modificar y eliminar.

Código:

```
    //Este método se ejecuta cada vez que se da clic sobre Agregar
            private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
     //Se revisa que el nombre no este vacío
     if(nom.getText().isEmpty())
    {
       JOptionPane.showMessageDialog(null,"Campos Vacios"); //Se hay vacíos entonces mostrara este mensaje
    }
    else{
    //De lo contrario se creara el siguiente objeto para las opciones del siguiente mensaje
      Object [] opciones ={"Aceptar","Cancelar"};
   //Se muestra el siguiente mensaje de selección
    int eleccion = JOptionPane.showOptionDialog(rootPane,"Estas seguro de agregar un nuevo registro","Mensaje de
Confirmacion",
    JOptionPane.YES_NO_OPTION,
    //Se agregan las opciones ingresadas en el objeto
    JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");
    //Si se selecciona la opción de Si en el mensaje de selección
```

```
     if(eleccion == JOptionPane.YES_OPTION)
     {
     try{
     //Mete a una cadena lo que se tiene en el nombre para realizar una consulta
     String nomb=nom.getText();
     //Se establece conexión con la base de datos
     Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
     St=Con.createStatement();
      //Se revisa ci ya existe un registro con ese nombre
      Rs = St.executeQuery("SELECT nom_suc FROM Sucursal WHERE nom_suc ="'+nomb+'"");
        if(Rs.last()){
      //Si existe mostrara el siguiente mensaje
            JOptionPane.showMessageDialog(null,"Ya existe un registro con ese nombre");
        }
        else{
      //Si no existe entonces realiza la siguiente consulta donde agrega la Sucursal ingresada
        String query="CALL NuevaSuc("'+nom.getText()+'");";
        St.executeUpdate(query);
      //Se notifica al usuario que el registro se agrego exitosamente
        JOptionPane.showMessageDialog(null,"Registro agregado exitosamente");
      //Se llama al método cargardatosTabla con la consulta de seleccionar todo de la tabla sucursal en la base de datos
        cargardatosTabla("select * from Sucursal");
      //Se agrega al autocompleta el nombre de las sucursales
        autocompleta.addItem(nom);
      //Se limpian los campos de texto de la interfaz
        nom.setText("");
        id.setText("");
        busqueda.setText("");
     //Se revisa si los botones de modificar y eliminar están activos
     if(modificar.isEnabled() && eliminar.isEnabled())
     {
    //Si es así entonces se desactivan
         modificar.setEnabled(false);
         eliminar.setEnabled(false);
     }
        }
     //Se cierra la conexión con la base de datos
     St.close();
     Rs.close();
     Con.close();
     //Se remueven los elementos del autocompleta
     autocompleta.removeAllItems();

     autocompletaDatos();
    //Si hay alguna excepción esta se atrapara aquí
            } catch (Exception e){JOptionPane.showMessageDialog(null,e);}
}
```

Botón Consultar Sucursal



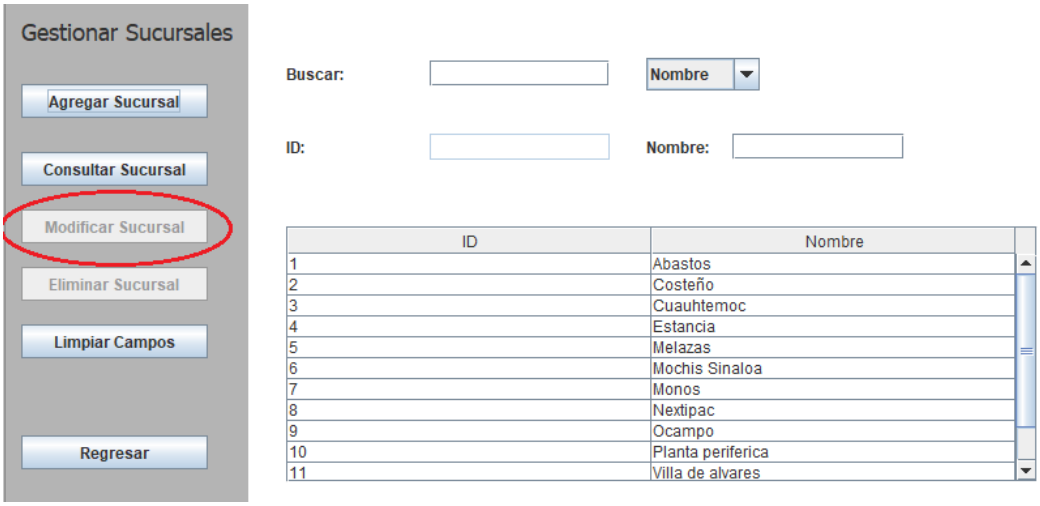
La función de botón consultar sucursales es la de llamar al método "consultarActionPerformed" el cual se ejecuta cada vez que se da clic en consultar, este método obtiene el texto que se introdujo en el campo de búsqueda y llama al método de "consultar" enviando ese parámetro, este comprobara si el parámetro está vacío, si lo está entonces mostrara el mensaje de "Campo de búsqueda vacío", de lo contrario establecerá conexión con la base de datos, declarara una variable del tipo String para la query y otra del tipo bolean para revisar si existe el registro a consultar, se determina de qué manera se está realizando la búsqueda, si es por el nombre o por ID, se realiza la consulta correspondiente en cada caso, se revisa que se obtenga algún registro, en caso de no obtenerlo se mostrara el mensaje de "No existe el registro", en caso contrario se obtendrán los datos de dicho registro y se pondrán en los campos de texto correspondientes, se cerrara conexión con la base de datos, se activaran los botones de modificar y eliminar y por último se limpiara el campo de texto de la búsqueda.

Código:

```
//Este método se llama cada vez que se da clic en consultar
//Su función es consultar un registro en la base de datos del sistema y cargar sus datos
private void consultarActionPerformed(java.awt.event.ActionEvent evt) {
        //Se obtiene lo que se tiene en el campo de texto de búsqueda
        String parametro=busqueda.getText();
       //Se llama al método consultar con el parámetro que se obtuvo del campo de texto
        consultar(parametro);
    }
   //Se llama cada vez que se realiza una consulta
   public void consultar(String parametro){
     //Revisa si el parámetro de entrada está vacío
     if(parametro.isEmpty())
       //Si está vacío entonces mostrara el mensaje de "Campo de búsqueda vacío"
       {JOptionPane.showMessageDialog(null,"Campo de busqueda vacio");}
       else{
```

```
    try{
      //De lo contrario se establecerá conexión con la base de datos del sistema
      Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
      St=Con.createStatement();
      //Se declara una variable la cual se utilizara para una consulta
      String query="";
     //Se declara una variable del tipo bolean se utilizara para comprobar si existe el registro o no
     boolean existe;
     //Se revisa si se tiene seleccionado en el ComboBox el nombre o el ID
     if(combobusqueda.getSelectedItem()=="Nombre"){
     //Si se tiene seleccionado el nombre se realiza la siguiente consulta
       Rs = St.executeQuery("CALL SucNom('"+parametro+"');");
     }
     else if(combobusqueda.getSelectedItem()=="ID"){
       //De lo contrario se llama al método de quitar espacios enviando el parámetro
       parametro=quitaEspacios(parametro);
       //Se recibe el parámetro ya modificado y se le saca su valor en digito para poder realizar la consulta por ID
       Rs = St.executeQuery("CALL BuscarSuc("+Integer.parseInt(parametro)+");");
     }

          //Revisa si existe el registro en la base de datos
          if(Rs.last()){existe=true;}
          else{
          //Si la variable boolean esta false el registro no existe y mostrara el siguiente mensaje "No existe el registro"
            JOptionPane.showMessageDialog(null,"No existe el registro");
            existe=false;
          }

          if(existe==true){
              //De lo contrario existe por lo que ahora revisa de qué manera está realizando la consulta
              if(combobusqueda.getSelectedItem()=="ID"){ //busqueda por id
               //Ejecuta la búsqueda por ID si se tiene seleccionado el ID en el ComboBox
               //query="Select * FROM Sucursal where ID_Suc='"+Integer.parseInt(parametro)+"';";
                 query="CALL BuscarSuc("+Integer.parseInt(parametro)+");";
               }
              else if(combobusqueda.getSelectedItem()=="Nombre") //busqueda por nombre
               {
                //Si tiene seleccionado el nombre la realizara por el nombre
                //query="Select * FROM Sucursal where nom_suc='"+parametro+"';";
                 query="CALL SucNom('"+parametro+"');";
               }
               //Se ejecuta la query
               Rs=St.executeQuery(query);
               //Se llama al método cargar datos para cargar los datos en las tablas
               cargardatos();
               //Se activan los botones de modificar y eliminar
               modificar.setEnabled(true);
               eliminar.setEnabled(true);
               //Se limpia el campo de texto para la búsqueda
               busqueda.setText("");
          }
         //Se cierra la conexión con la base de datos
          St.close();
          Rs.close();
          Con.close();
        //Si ocurre una excepción esta se atrapa aquí
        } catch (Exception e){JOptionPane.showMessageDialog(null,e);}
    }
}
```

Boton Modificar Sucursal



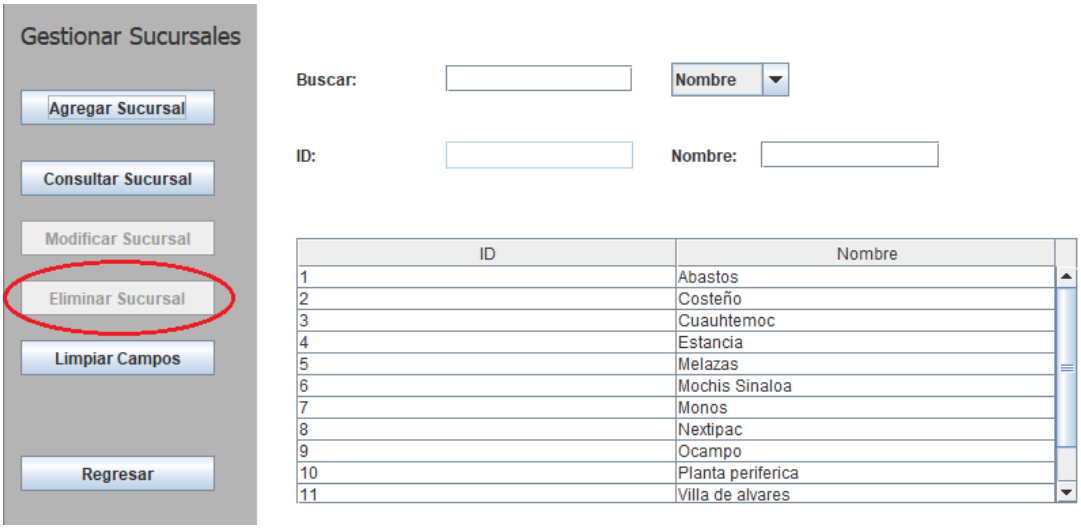
Método modificarActionPerformed

El siguiente método es utilizado para modificar una sucursal, que se tiene en la base de datos, solo se puede modificar el nombre de la sucursal y se debe de haber realizado la consulta de la sucursal para poder modificarla.

```
   private void modificarActionPerformed(java.awt.event.ActionEvent evt) {     //Botón modificar sucursal
        if(nom.getText().isEmpty()) // Si campo nom esta vacío
    {
       JOptionPane.showMessageDialog(null,"Campos Vacios"); // Muestra mensaje de campos vacíos
    }
    else{ //Sino
Object [] opciones ={"Aceptar","Cancelar"}; //Crea un objeto con Aceptar y Cancelar
     int eleccion = JOptionPane.showOptionDialog(rootPane,"Estas seguro de modificar este registro","Mensaje de
Confirmacion",
     JOptionPane.YES_NO_OPTION,
     JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar"); // Se crea el mensaje de confirmación
     if(eleccion == JOptionPane.YES_OPTION) // Si la selección fue aceptar
     {
     try{
     Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); // Se realiza conexión a la base de
datos
     St=Con.createStatement();
     //String query="update Sucursal set nom_suc='"+nom.getText()+"' where
ID_Suc='"+Integer.parseInt(id.getText())+"';";
     String query="CALL ModiSuc('"+nom.getText()+"','"+Integer.parseInt(id.getText())+"');"; // Se llama al
procedimiento y se manda lo que contiene el campo nombre y la conversión a entero de lo que contiene el campo id
     St.executeUpdate(query);
     St.close();
     Con.close(); //Se cierra la conexión con la base de datos
     cargardatosTabla("select * from Sucursal");// Se invoca el método cargardatosTabla con la selección de toda la tabla
Sucursal
     nom.setText(""); // Se vacía el campo nom
     id.setText(""); // Se vacía el campo id
     JOptionPane.showMessageDialog(null,"Registro modificado exitosamente"); // Se muestra el mensaje de éxito
     modificar.setEnabled(false); // Se desactiva el botón modificar
     eliminar.setEnabled(false); // Se desactiva el botón eliminar
     autocompleta.removeAllItems(); // Se quitan todos los items de autocompleta
     autocompletaDatos(); // Se invoca el método aut}
```

```
        } catch (Exception e){JOptionPane.showMessageDialog(null,e);} // Si ocurre un error muestra el mensaje  error y cual error fue
    }
    }
    }
```

Botón Eliminar Sucursal



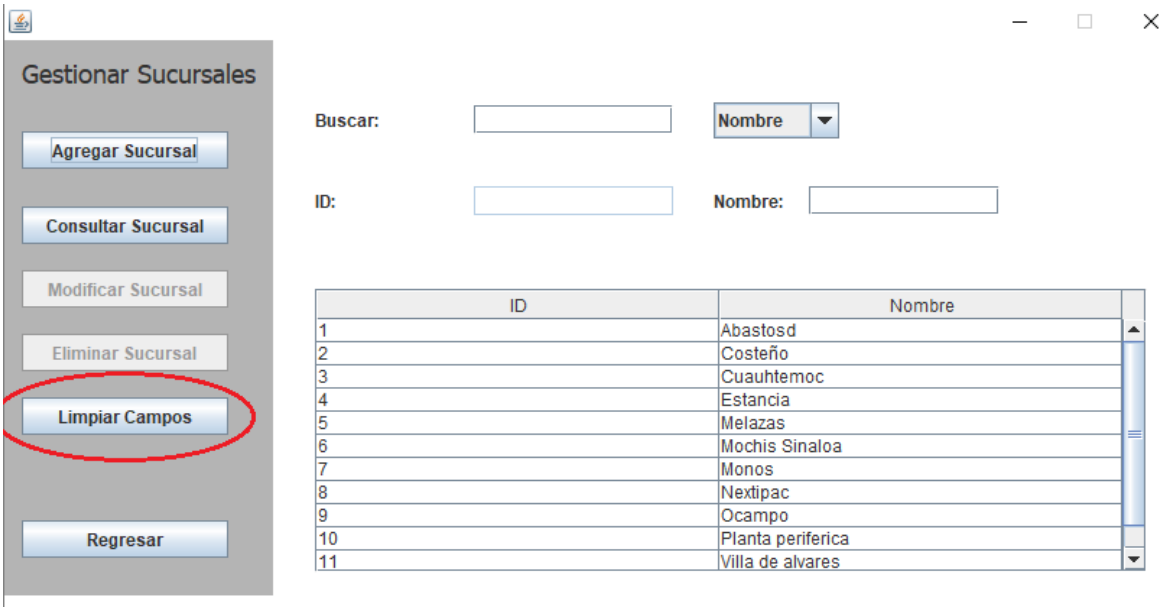
Método eliminarActionPerformed

El siguiente método es utilizado para eliminar una sucursal que se tiene en la base de datos, solo se puede eliminar la sucursal si no está en uso por algún personal.

```
    private void eliminarActionPerformed(java.awt.event.ActionEvent evt) {      //Botón de eliminar sucursal
            if(nom.getText().isEmpty()) //Si el campo nom esta vacío
     {
        JOptionPane.showMessageDialog(null,"Campos Vacios"); //Muestra el mensaje de campos vacíos
     }
     else{ // Sino
     Object [] opciones ={"Aceptar","Cancelar"}; //Crea un objeto con Aceptar y Cancelar
     int eleccion = JOptionPane.showOptionDialog(rootPane,"Estas seguro de eliminar este registro","Mensaje de Confirmacion",
     JOptionPane.YES_NO_OPTION,
     JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");// Se crea el mensaje de confirmación
     if(eleccion == JOptionPane.YES_OPTION) // Si la opción seleccionada fue  Aceptar
     {
     try{
     Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); // Realiza conexión con la base de datos
     St=Con.createStatement();
     String query="CALL EliminarSuc('"+Integer.parseInt(id.getText())+"');"; // Invoca el procedimiento de eliminar sucursal mandando la conversión del campo id a entero
     St.executeUpdate(query);
     St.close();
     Con.close(); // Cierra conexión con la base de datos
     cargardatosTabla("select * from Sucursal"); // Invoca el método cargardatosTabla  mandando la selección de todo lo que tiene la tabla sucursal
     nom.setText(""); // Se vacía el campo nom
     id.setText(""); // Se vacía el campo id
```

```
    busqueda.setText(""); // Se vacía el campo de búsqueda
    modificar.setEnabled(false); // Se desactiva el botón modificar
    eliminar.setEnabled(false); // Se desactiva el botón eliminar
    autocompleta.removeAllItems(); // Se eliminan todos los items de autocompleta
    autocompletaDatos(); // Se invoca el método autocompleta
          } catch (Exception e){JOptionPane.showMessageDialog(null,e);} // Si ocurre un error muestra cual error fue
    }
}
    }
```

Botón Limpiar Campos



jButton2ActionPerformed

El siguiente método es utilizado para limpiar los campos de texto cuando el botón de "Limpiar Campos" es presionado.

```
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {    //Botón Limpiar campos
    nom.setText(""); //Se vacia el campo nom
    id.setText(""); //Se vacia el campo id
    busqueda.setText(""); //Se vacia el campo de busqueda
        if(modificar.isEnabled() && eliminar.isEnabled()) //Si el boton de modificar y eliminar estan activos
    {
       modificar.setEnabled(false); //Desactiva el botón de modificar
       eliminar.setEnabled(false);//Desactiva el botón de elimiar
    }
        cargardatosTabla("select * from Sucursal"); //Invoca el metodo cargardatosTabla con todo lo que se tenga en la
tabla sucursal
    }
```

Panel de control

Método ControlPanel

El siguiente método es utilizado para crear el método constructor de la clase, en este método se define el color de fondo, se activa o desactiva el cambiar tamaño de ventana, activar o desactivar módulos dependiendo del usuario.

```
public ControlPanel() { // Se crea metodo constructor
      initComponents();
      getContentPane().setBackground(new java.awt.Color(205,92,92)); // Se define el color de fondo
      setLocationRelativeTo(null);
      setResizable(false); // Se desactiva el poder cambiar el tamaño de la ventana
      this.setTitle("Menu principal - Usuario: "+Globales.User); //Para indicar con que usuario se ingresó al sistema

      ((JPanel)getContentPane()).setOpaque(false);
      ImageIcon uno=new ImageIcon(this.getClass().getResource("/Imagenes/claro.jpg"));   //Se carga el icono de fondo
      JLabel fondo= new JLabel();
      fondo.setIcon(uno);
      getLayeredPane().add(fondo,JLayeredPane.FRAME_CONTENT_LAYER);
      fondo.setBounds(0,0,uno.getIconWidth(),uno.getIconHeight());
      if(Globales.User.equals("Administrador")||Globales.User.equals("root")){ //Si el usuario es igual a administrador o a
root
        nuevousuario.setEnabled(true); // Activa el botón usuario
      }
      else nuevousuario.setEnabled(false); // Sino desactívalo
   }
```

Botón Gestionar Personal

Método jButton5ActionPerformed

El siguiente método es utilizado para ingresar al módulo Gestionar personal cuando se presiona el botón de "Gestionar Personal".

```
    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {      //Botón Gestionar Personal
        this.dispose();
        new Personal().setVisible(true);        //Se cambia la ventana de Personal a visible
}
```

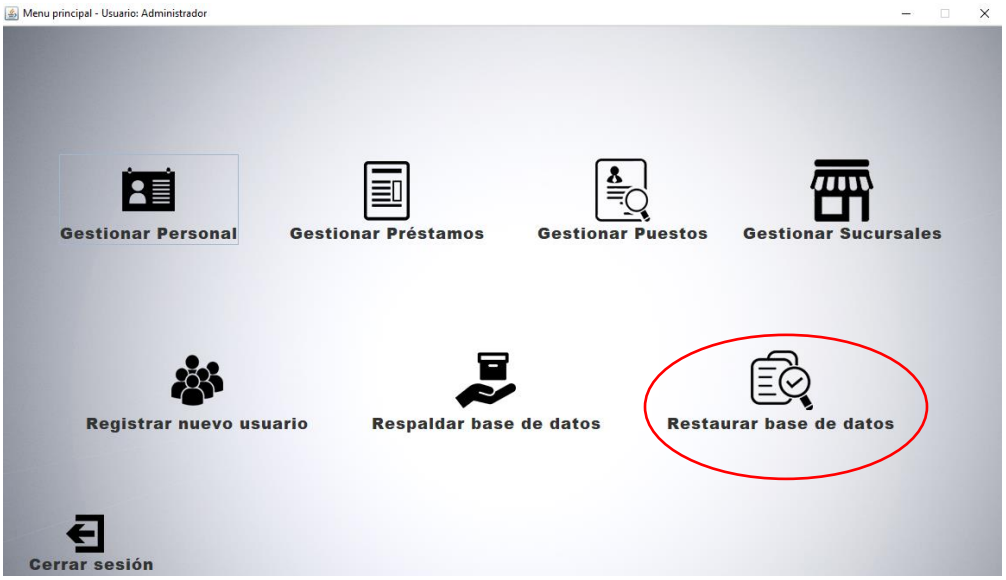
Botón Gestionar Préstamos

Método gestionprestamosActionPerformed

El siguiente método es utilizado para ingresar al módulo Gestionar Préstamos cuando se presiona el botón de “Gestionar Préstamos”.

```
private void gestionprestamosActionPerformed(java.awt.event.ActionEvent evt) {  //Botón gestionar prestamos
    new GestionarPrestamos().setVisible(true); //Se cambia la ventana de Préstamos a visible
    this.dispose();
}
```

Botón Restaurar Base de Datos

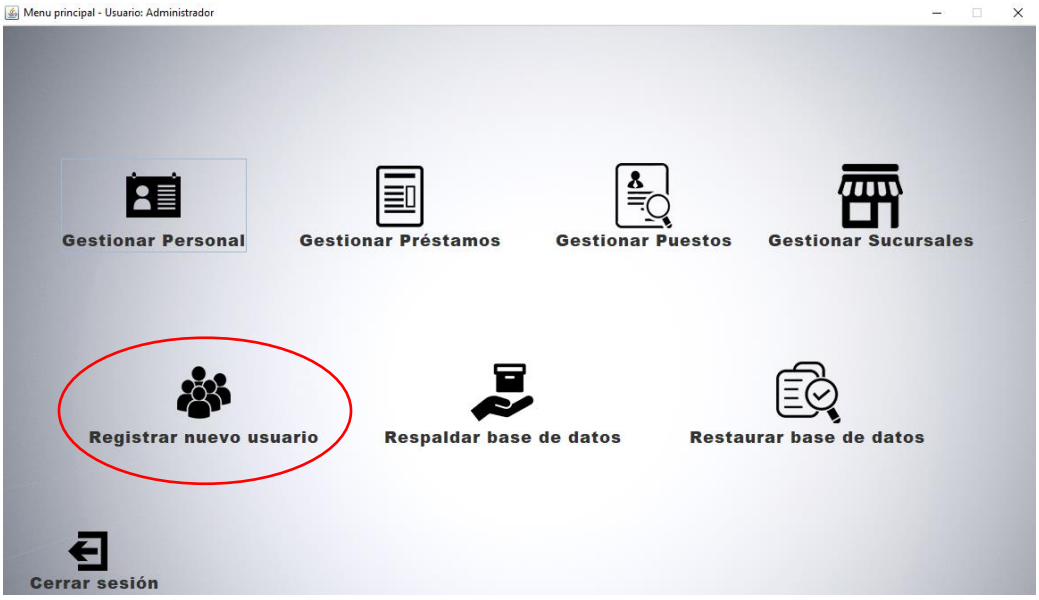


Método restauracionActionPerformed

El siguiente método es utilizado para ingresar al módulo Restaurar base de datos cuando se presiona el botón de “Restaurar base de datos”.

```
private void restauracionActionPerformed(java.awt.event.ActionEvent evt) {      // Botón Restaurar Base de Datos
    new Restauracion().setVisible(true); // Se cambia a visible la interface de Restauración
}
```

Botón Registrar Usuario



Método nuevousuarioActionPerformed

El siguiente método es utilizado para ingresar al módulo Registrar nuevo usuario cuando se presiona el botón de "Registrar nuevo usuario".

```
private void nuevousuarioActionPerformed(java.awt.event.ActionEvent evt) {     // Botón para nuevo usuario
 this.dispose();
  new RegistrarUsuario().setVisible(true);  // Se cambia la interface de RegistrarUsuario a visible
}
```

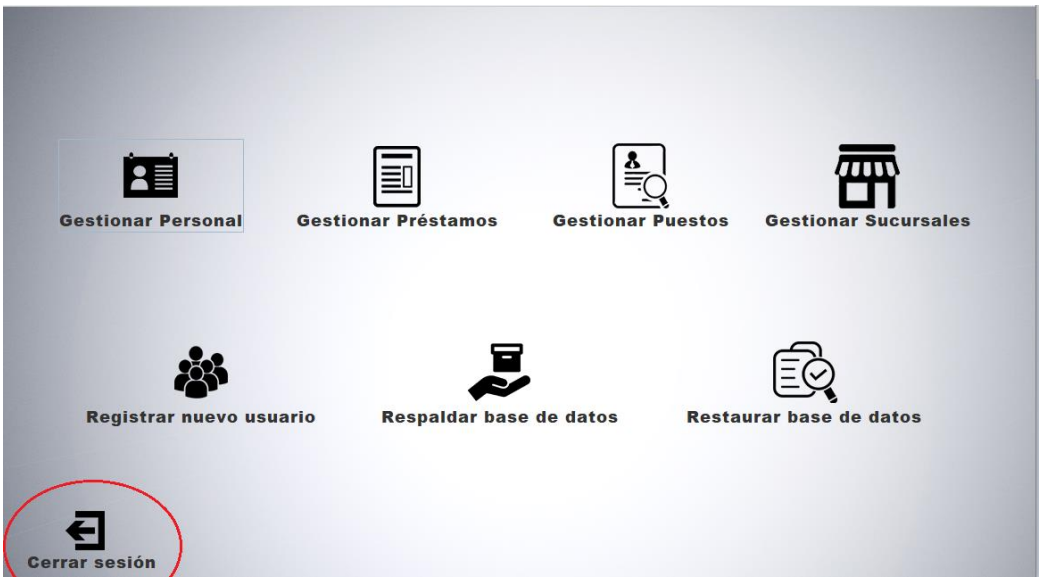
Botón Gestionar Puestos

Método gestionpuestoActionPerformed

El siguiente método es utilizado para ingresar al módulo Gestionar Puestos cuando se presiona el botón de “Gestionar Puestos”.

```
private void gestionpuestoActionPerformed(java.awt.event.ActionEvent evt) {     // Botón de Gestionar Puesto
new PuestoD().setVisible(true); // Se cambia la interface de PuestoD a visible
this.dispose();
}
```

Botón Cerrar Sesión

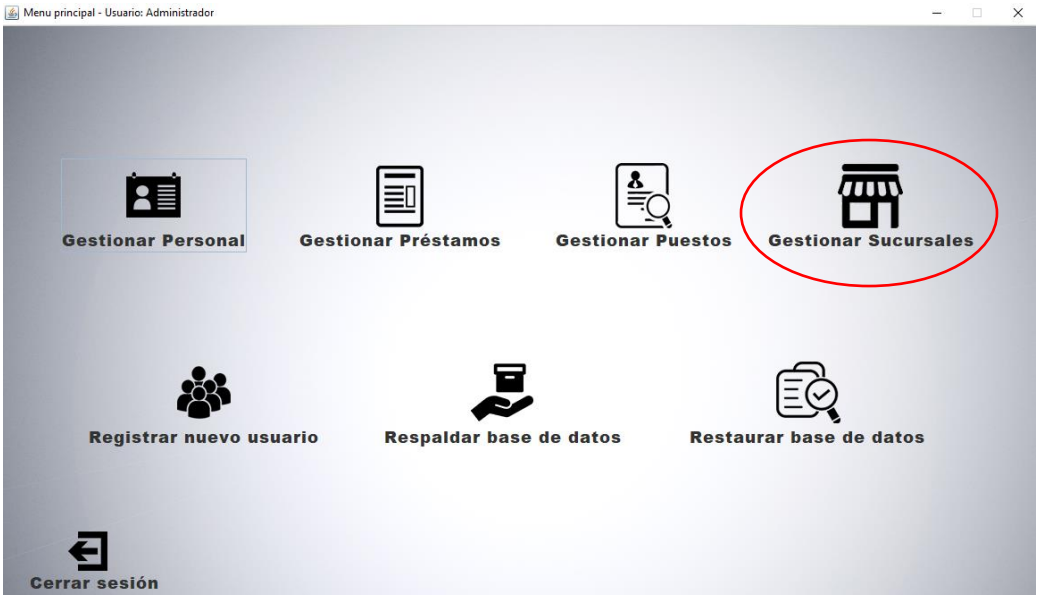


Método LogOutButtonActionPerformed

El siguiente método es utilizado para regresar al login cuando se presiona el botón de “Cerrar sesión”.

```
private void LogOutButtonActionPerformed(java.awt.event.ActionEvent evt) {     // Botón de regresar a Login
    this.dispose();
    new Login().setVisible(true);  //Se cambia la interfaz de Login a visible
}
```

Botón Gestionar Sucursales



Método gestionsucursalActionPerformed

El siguiente método es utilizado para ingresar al módulo Gestionar Sucursales cuando se presiona el botón de "Gestionar Sucursales".

```
private void gestionsucursalActionPerformed(java.awt.event.ActionEvent evt) {    //Botón gestionar Sucursales
    new Sucursales().setVisible(true); // Se cambia la interfaz de Sucursales a visible
    this.dispose();
}
```

Ingresar Usuario y Contraseña

Método Login

El siguiente método es utilizado para crear el método constructor de Autentificación de usuarios, en este se define algunos atributos como color de fondo, donde aparecerá la ventana, entre otros.

```
public Login() {// Método constructor de Login
    initComponents();
```

```
    setLocationRelativeTo(null);//para que aparezca centrado
    getContentPane().setBackground(new java.awt.Color(255,255,255));//change color frame
    setLayout(null);

    ((JPanel)getContentPane()).setOpaque(false);
    ImageIcon uno=new ImageIcon(this.getClass().getResource("/Imagenes/claro.jpg")); //Se carga el icono de fondo
    JLabel fondo= new JLabel();
    fondo.setIcon(uno); //A la etiqueta fondo se manda el icono
    getLayeredPane().add(fondo,JLayeredPane.FRAME_CONTENT_LAYER);
    fondo.setBounds(0,0,uno.getIconWidth(),uno.getIconHeight());
}
```

Botón Ingresar

Método jButton1ActionPerformed

El siguiente método es utilizado para cuando sea presionado tome el usuario y la contraseña ingresados, e ingrese al sistema si estos son válidos y existen en la base de datos.

```
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {   //Botón ingresar
       Globales.Url="jdbc:mysql://localhost/dsmafornu";  //Se iguala la variable url
       Globales.User=usuarioTexto.getText(); //Se iguala la variable usuario a lo que contiene el campo de texto usuario
       Globales.Pass=contraseñaTexto.getText(); // Se iguala la variable contraseña a lo que contiene el campo de texto
contraseña
      if(Globales.User.isEmpty()||Globales.Pass.isEmpty()){ //Si el campo usuario o  contraseña están vacíos
         JOptionPane.showMessageDialog(null,"Campos vacios"); //Muestra el mensaje
      }else{ //Sino
         try {
            Class.forName("com.mysql.jdbc.Driver");//cargar el controlador
            java.sql.Connection con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass); //Se realiza
la conexión a la base de datos
            con.close();//nos aseguramos de cerrar la conexión, no es muy necesario pero es recomendable
            new ControlPanel().setVisible(true); // Se cambia la ventana de ControlPanel a visible
            this.dispose();
         }
         catch (ClassNotFoundException ex) { //Si se produce un error
           JOptionPane.showMessageDialog(this,ex.getMessage()); //Muestra cual error fue
         }catch(SQLException ex){ //Si se produce un error en SQL
           JOptionPane.showMessageDialog(this,ex.getMessage()); //Muestra cual error fue
         }

      }//else
   }
```

Botón Cancelar

Método jButton2ActionPerformed

El siguiente método es utilizado para cuando sea presionado cierre el sistema.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {    //Botón cancelar
    cerrarsistema();  //Invoca el método cerrarsistema
}
```

Botón Cerrar

Método cerrarBotonAgregar

El siguiente método es utilizado para cuando sea presionado cierre el sistema.

```
private void cerrarBotonAgregar(java.awt.event.ActionEvent evt) {    //Botón de X para salir
    cerrarsistema(); //Se invoca el método cerrarsistema
}
```

Botón Minimizar

Método cerrarBotonAgregar

El siguiente método es utilizado para cuando sea presionado minimice el sistema.

```
private void minimizarBotonAgregar(java.awt.event.ActionEvent evt) {     //Botón de minimizar ventana
    this.setExtendedState(ICONIFIED); //Se cambia el estado de la ventana
}
```

Método formMousePressed

El siguiente método es utilizado para obtener la posición en pantalla de la ventana.

```
private void formMousePressed(java.awt.event.MouseEvent evt) {
    x=evt.getX(); //Va obteniendo e igualando X a la posición de x
```

```
    y=evt.getY(); //Va obteniendo e igualando Y a la posición de y
  }
```

Método formMouseDragged

El siguiente método es utilizado para mover a otra posición en pantalla la ventana.

```
  private void formMouseDragged(java.awt.event.MouseEvent evt) {   //Método para mover la ventana a la posición obtenida
    this.setLocation(this.getLocation().x +evt.getX() -x,this.getLocation().y +evt.getY() -y); //Mueve la ventana a los valores de x & y
  }
```

Método usuarioTextoKeyTyped

El siguiente método es utilizado para mandar un sonido de error y eliminar el carácter ingresado cuando este no es válido.

```
  private void usuarioTextoKeyTyped(java.awt.event.KeyEvent evt) {    // Validar que no ingrese caracteres inválidos en el campo de usuario
    String NOMBRE=usuarioTexto.getText(); //Se crea y se iguala la variable NOMBRE al contenido del campo de texto usuarioTexto
     char k=evt.getKeyChar(); //Se crea y se iguala la variable k a la tecla presionada
    if(Character.isLetter(k)||Character.isDigit(k)){ //Si la tecla presionada es letra, o si la letra presionada es digito
     if(NOMBRE.length()>0){   //Si el tamaño de NOMBRE es mayor a cero
       if(NOMBRE.length()==15){ //Si el tamaño de NOMBRE es igual a 15
        getToolkit().beep(); //Invoca sonido de error de Windows
        evt.consume(); //Borra el carácter ingresado
       }
     }
     else if(NOMBRE.isEmpty()){ //Si NOMBRE esta vacío
       if(k==' '||k=='.'){ //Si la tecla presionada es espacio o punto
        getToolkit().beep();//Invoca sonido de error de Windows
        evt.consume();   //Borra el carácter ingresado
       }
     }
    }else{
        evt.consume();   //Borra el carácter ingresado
    }
  }
```

Método contraseñaTextoKeyTyped

El siguiente método es utilizado para mandar un sonido de error y eliminar el carácter ingresado cuando este no es válido.

```
  private void contraseñaTextoKeyTyped(java.awt.event.KeyEvent evt) {    //Validar que no ingrese caracteres inválidos en el campo de contraseña
    char k=evt.getKeyChar(); // Se crea la variable k y se iguala a la tecla presionada
     String NUMERO=contraseñaTexto.getText(); //Se crea la variable NUMERO y se iguala a lo que contiene el campo contraseña
    if(Character.isDigit(k)||Character.isLetter(k)){ // Si el carácter ingresado es digito o letra
     if(k==' '){ //Si la tecla presionada es espacio
       getToolkit().beep(); //Invoca sonido de error de Windows
       evt.consume();    ////Borra el carácter ingresado
     }
     else if(NUMERO.length()>0){   //Si el tamaño de NUMERO es mayor a cero
       if(NUMERO.length()==15){ //Si el tamaño de NUMERO es igual a 15
        getToolkit().beep();//Invoca sonido de error de Windows
        evt.consume();////Borra el carácter ingresado
       }
     }
```

```
    }else{     //Sino
        evt.consume(); ////Borra el carácter ingresado
    }
}
```

Método Cerrarsistema

Este método es utilizado para cerrar el sistema completamente.

```
public void cerrarsistema(){ //Método para cerrar sistema
System.exit(0); //Se cierra el sistema
}
```

Modulo Registrar Usuario

Se realizan las importaciones de las librerías necesarias para poder desarrollar de manera eficiente los métodos que se usan en el módulo.

```
//importaciones de librerías necesarias
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
```

Se declaran las variables globales que se usaran en la mayoría de los métodos.

```
Connection Con; //Conexión
Statement St;   //Ejecutar comandos
ResultSet Rs;   //Resultado de la consulta
```

Método Registrar Usuario

Este es el método constructor de la clase del módulo Registrar Usuario. En este método se inicializan los componentes necesarios para el correcto funcionamiento del módulo y sus métodos así como también se encarga de definir ciertos aspectos de la venta.

```
//declaración de variables globales
static int s,i,u,d;
//método constructor de la clase
public RegistrarUsuario() {
    //se ajusta el encabezado de la ventana
    this.setTitle("Registrar usuario - Usuario: "+Globales.User);
    //se inicializan los componentes
    initComponents();
    setLocationRelativeTo(null);//para que aparezca centrado
    s=0;i=0;u=0;d=0;
    //no se permite reajustar la ventana
    setResizable(false);
    //el cursor se pone por defecto en el campo de texto del nombre
    nombre.requestFocus();
}
```

Método nombreKeyPressed

Este método se manda a llamar cada vez que se quiere cambiar el cursor del campo de texto nombre al de contraseña.

```
private void nombreKeyPressed(java.awt.event.KeyEvent evt) {
    //si el cursor está en el campo de texto del nombre
    //y se presiona enter el cursor se mueve al campo de la contraseña
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
      contra.requestFocus();
    }
}
```

Método contraKeyPressed

Este método se manda a llamar cada vez que se quiere cambiar el cursor del campo de texto de la contraseña al de repite la contraseña o al del nombre.

```
private void contraKeyPressed(java.awt.event.KeyEvent evt) {
    //si el cursor esta en el campo de texto contra
    //y presiona enter el cursor se mueve al campo de contra2
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
      contra2.requestFocus();
    }
    //si lo contrario, si se presiona la flecha arriba el cursor se mueve al campo nombre
    else if(evt.getKeyCode()==KeyEvent.VK_UP){
      nombre.requestFocus();
    }
}
```

Método contra2KeyPressed

Este método se manda a llamar cada vez que se quiere cambiar el cursor del campo de texto de repite la contraseña.

```
private void contra2KeyPressed(java.awt.event.KeyEvent evt) {
    //si el cursor esta en el campo de texto contra
    //y se presiona el boton arriba el cursor se dirige al campo contra
```

```
    if(evt.getKeyCode()==KeyEvent.VK_UP){
      contra.requestFocus();
    }
}
```

Método nombreKeyTyped

Este método se manda a llamar cada vez que se inserta un carácter en el campo de texto nombre con la intención de solo aceptar los que sean válidos de acuerdo a la documentación.

```
private void nombreKeyTyped(java.awt.event.KeyEvent evt) {
    //se obtienen el caracter ingresado y los caracteres que
    //que ya estan en el campo de texto
    String NOMBRE=nombre.getText();
      char k=evt.getKeyChar();
     //solo se analizan letras y digitos
     if(Character.isLetter(k)||Character.isDigit(k)){
       //si el campo de texto no esta vacio
         if(NOMBRE.length()>0){
         //solo se permiten 15 caracteres
         if(NOMBRE.length()==15){
          getToolkit().beep();
          evt.consume();
        }
       }
       //si el campo de texto esta vacio
       //no se permiten espacios o puntos
       else if(NOMBRE.isEmpty()){
         if(k==' '||k=='.'){
           getToolkit().beep();
           evt.consume();
         }
       }
      }
      //si no es letra o digito no se acepta el caracter
      else{
          evt.consume();
      }
}
```

Método contraKeyTyped

Este método se manda a llamar cada vez que se inserta un carácter en el campo de texto contra con la intención de solo aceptar los que sean válidos de acuerdo a la documentación.

```
private void contraKeyTyped(java.awt.event.KeyEvent evt) {
    //se obtienen el carácter ingresado y los caracteres que
    //que ya están en el campo de texto
    char k=evt.getKeyChar();
    String NUMERO=contra.getText();
    //solo se analizan letras y dígitos
   if(Character.isDigit(k)||Character.isLetter(k)){
     //no se aceptan espacios
     if(k==' '){
     getToolkit().beep();
     evt.consume();
    }
    else if(NUMERO.length()>0){
      //solo se aceptan 15 caracteres
      if(NUMERO.length()==15){
```

```
        getToolkit().beep();
        evt.consume();
       }
     }
    }
    //no se acepta el carácter si no es letra o digito
    else{
        evt.consume();
    }
}
```

Método contra2KeyTyped

Este método se manda a llamar cada vez que se inserta un carácter en el campo de texto contra2 con la intención de solo aceptar los que sean válidos de acuerdo a la documentación.

```
private void contra2KeyTyped(java.awt.event.KeyEvent evt) {
     //se obtienen el carácter ingresado y los caracteres que
    //que ya están en el campo de texto
    char k=evt.getKeyChar();
    String NUMERO=contra2.getText();
    //solo se analizan letras y digitos
   if(Character.isDigit(k)||Character.isLetter(k)){
      //no se aceptan espacios
      if(k==' '){
      getToolkit().beep();
      evt.consume();
    }
     else if(NUMERO.length()>0){
       //solo se aceptan 15 caracteres
       if(NUMERO.length()==15){
        getToolkit().beep();
        evt.consume();
       }
     }
   }
   //no se acepta el carácter si no es letra o digito
   else{
       evt.consume();
   }
}
```

Botón Aceptar

Al presionar el botón “Aceptar” se manda a llamar a su método actionperformed el cual crea un nuevo usuario a partir de los parámetros que ingreso el usuario y a ves checa que la validación sea correcta, que no haya espacios en blanco, que no exista el registro y que las contraseñas coincidan.

```
   private void jButton3BotonAgregar(java.awt.event.ActionEvent evt) {
      //se valida que no haya campos vacíos
      if(nombre.getText().isEmpty() || contra.getText().isEmpty() ||
contra2.getText().isEmpty()){JOptionPane.showMessageDialog(null,"Campos vacios");}
      else{
        //se valida que ambas contraseñas sean iguales
       if(contra.getText().equals(contra2.getText())){

         //se le pregunta al usuario si está seguro de sus acciones
         Object [] opciones ={"Aceptar","Cancelar"};
         int eleccion = JOptionPane.showOptionDialog(rootPane,"¿Estas seguro de agregar un nuevo registro?","Mensaje
de Confirmacion",
         JOptionPane.YES_NO_OPTION,
         JOptionPane.QUESTION_MESSAGE,null,opciones,"Aceptar");

         //si el usuario selecciono que aceptar
         if (eleccion == JOptionPane.YES_OPTION)
         {
             try {
                //se raliza conexion con la base de datos
                Con=DriverManager.getConnection(Globales.Url,Globales.User,Globales.Pass);
                St=Con.createStatement();
                //se ejecutan los comandos necesarios para crear al nuevo usuario
                St.executeUpdate("CREATE USER'"+nombre.getText()+"'@localhost IDENTIFIED
BY'"+contra.getText()+"';");
                St.executeUpdate("GRANT ALL PRIVILEGES ON dsmafornu.* TO '"+nombre.getText()+"'@'localhost';");
                //St.executeUpdate("GRANT ALL PRIVILEGES ON *.* TO '"+nombre.getText()+"'@localhost REQUIRE
NONE WITH GRANT OPTION;");
                JOptionPane.showMessageDialog(null,"Registro agregado exitosamente");
                //se limpian los campos de texto
                nombre.setText("");
```

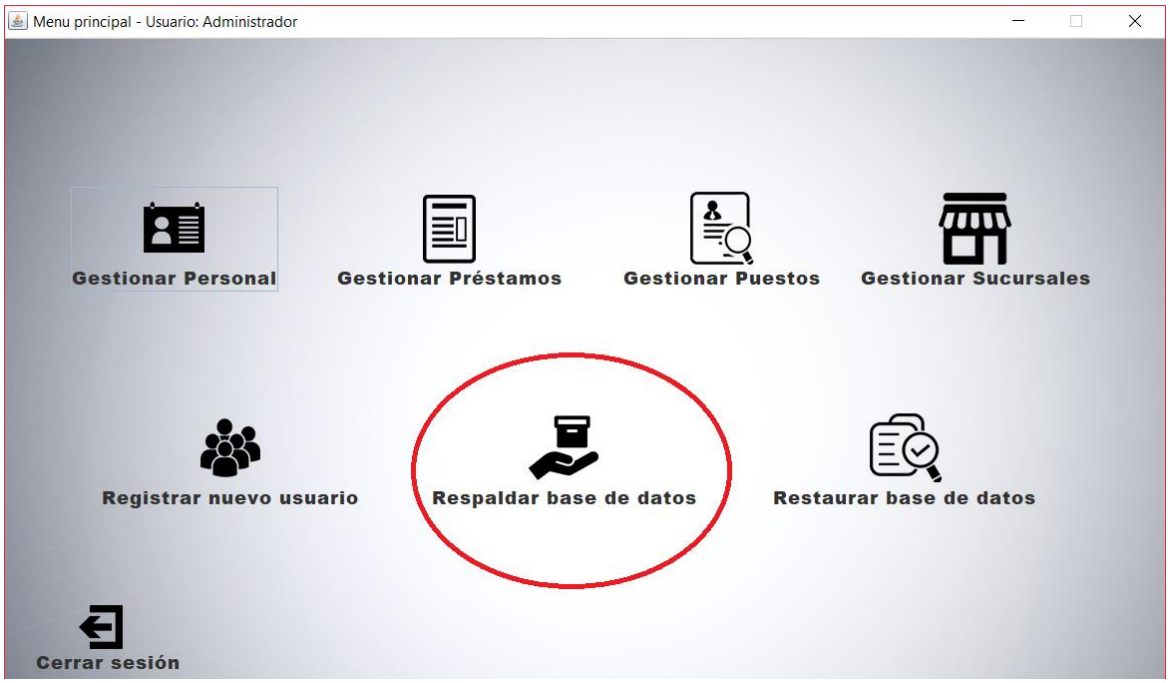
```
            contra.setText("");
            contra2.setText("");
          }  catch (Exception e) {
            //si hay algún error aquí se muestra
            JOptionPane.showMessageDialog(null,e);
            }

       }//if de confirmación
      }
      else{
         //se llega a esta condicion si las contraseñas no son iguales
       JOptionPane.showMessageDialog(null,"La contraseña debe coincidir");
      }
     }
}
```

Botón Cancelar

Al presionar el botón cancelar se manda a llamar a su método actionperformed el cual cierra la ventana de Registrar Usuario y abre el menú principal.

```
private void CancelarBotonAgregar(java.awt.event.ActionEvent evt) {
  //se cierra esta ventana y se abre la del menú principal
   this.dispose();
   new ControlPanel().setVisible(true);
}
```

Botón Respaldar base de datos

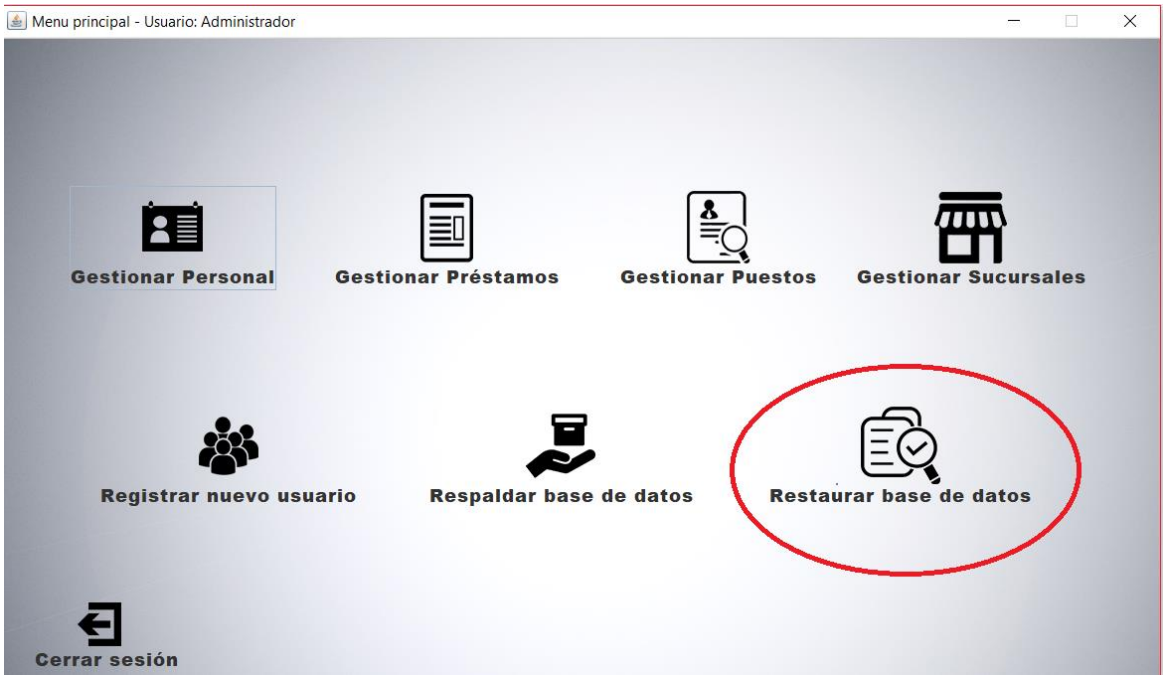


Al presionar el botón "Respaldar base de datos" se manda a llamar a su método actionperformed el cual crea una nueva ventana para poder seleccionar la ruta donde se quiere guardar el respaldo, una ves seleccionada la ruta este método genera un archivo .sql el cual contiene a manera de código las tablas y registros que se encuentran en la base de datos.

```
private void respaldobasedatosActionPerformed(java.awt.event.ActionEvent evt) {

     JFrame respaldo=new JFrame();//creo un jframe
    JFileChooser fileChooser = new JFileChooser();//un objeto del tipo jfilechooser
    fileChooser.setDialogTitle("Elige la ubicación para tu respaldo"); //con esto le pongo el título al frame

    //creo una variable int para obtener si le da al botón guardar
    int userSelection = fileChooser.showSaveDialog(respaldo);//despues le digo que mi objeto filechooser pertenece al
tipo guardar y esta en el frame respaldo
    File fileToSave = fileChooser.getSelectedFile();
    String ruta=fileToSave.getAbsolutePath();//obtengo la ruta de la carpeta

     try {
        Runtime runtime=Runtime.getRuntime();
        FileWriter fw=new FileWriter(ruta);
         //se ejecuta el comando para realizar el respaldo
        //para xamp
        //Process p =runtime.exec("C:\\xampp\\mysql\\bin\\mysqldump --opt --password= --user=root --database=
dsmafornu -R");
        //para mysql essential
        Process p =runtime.exec("C:\\mysql\\bin\\mysqldump --opt --password= --user=root --database= dsmafornu -R");

        //se preparan variables de lectura y escritura
        InputStreamReader is=new InputStreamReader(p.getInputStream());
```

```
        BufferedReader br=new BufferedReader(is);

        //se hace el respaldo a base de escritura
        String line;
        while((line=br.readLine())!=null){
         fw.write(line + "\n");
        }

        //se cierran las variables de escritura
        fw.close();
        is.close();
        br.close();
        JOptionPane.showMessageDialog(null,"Respaldo creado exitosamente");
    } catch (Exception e) {
        //si hay algún error aquí se muestra
        JOptionPane.showMessageDialog(null,e);
    }
}
```

Botón Restaurar base de datos



Al presionar el botón “Restaurar base de datos” esta llama a su método actionperformed el cual abre una nueva ventana para selección de archivos y permite seleccionar el respaldo por el usuario para restaurar la base de datos por medio de sentencias en la terminal que son directamente ejecutadas desde java.

```
private void restauracionActionPerformed(java.awt.event.ActionEvent evt) {
    //se invoca a la ventana restauración
     new Restauracion().setVisible(true);
  }

private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    //se crean los objetos necesarios para tomar la ruta elegida
    JFileChooser selector=(JFileChooser)evt.getSource();
    String comando=evt.getActionCommand();

    //si el usuario presiono aceptar
    if(comando.equals(JFileChooser.APPROVE_SELECTION)){
       File seleccionado=selector.getSelectedFile();

       try {
        //se ejecuta el comando para realizar el respaldo
        //original
        //Process p = Runtime.getRuntime().exec("C:\\xampp\\mysql\\bin\\mysql --user=root --password= dsmafornu");
        Process p = Runtime.getRuntime().exec("C:\\mysql\\bin\\mysql --user=root --password= dsmafornu");
        //se crean las variables para la lectura y escritura
        OutputStream os = p.getOutputStream();
        FileInputStream fis = new FileInputStream(seleccionado.getAbsolutePath());

        //se leen y se escriben las lineas del respaldo seleccionado
        //para restaurar la base de datos
        byte[] buffer = new byte[1000];

        int leido = fis.read(buffer);
         while (leido > 0) {
            os.write(buffer, 0, leido);
            leido = fis.read(buffer);
        }

        os.flush();
        os.close();
        fis.close();

        JOptionPane.showMessageDialog(null,"Base de datos restaurada exitosamente");
        this.dispose();
    } catch (Exception e) {
        //se muestran los errores
        JOptionPane.showMessageDialog(null,e);
    }
    }
    else if(comando.equals(JFileChooser.CANCEL_SELECTION)){
      //si se presiona el botón de cancelar se cierra la ventana de restauración
        this.dispose();
    }
}
```