EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Gestión de una tabla

En la hoja guiada anterior se vio como mostrar el contenido de una tabla de la base de datos en un JTable de la aplicación java.

Se puede aprovechar esta idea de forma que todas las operaciones que realicemos sobre una tabla se vean inmediatamente reflejadas en el JTable de la aplicación java.

Por ejemplo, si eliminamos un registro sería interesante que automáticamente viéramos la tabla actualizada en el JTable.

En esta hoja guiada se mejorará la aplicación de la hoja guiada anterior, dotándola de opciones para la gestión de la tabla *Trabajadores*.

EJERCICIO GUIADO Nº 1

1. Abrir la aplicación de la hoja guiada anterior.

2. El primer objetivo será hacer que al ejecutar el programa aparezca automáticamente el contenido de la tabla *Trabajadores* en el JTable. Para ello, realice los siguientes cambios en el código del programa:

```
/** Creates new form ventanaprincipal */
public ventanaprincipal() {
    initComponents();
    PrepararBaseDatos();
    PrepararTabla();
    MostrarTrabajadores();
}
```

Añade una llamada a un método *MostrarTrabajadores*, en el constructor del programa.

3. Crea el método *MostrarTrabajadores*:

```
void MostrarTrabajadores() {

}
```

4. Copia en él el código del *actionPerformed* del botón *btnTrabajadores*:

```
void MostrarTrabajadores() {
    String fecha;
    try {
        //Extraer datos de la tabla y almacenarlos en el resultset
        ResultSet r = sentencia.executeQuery("select * from trabajadores");
        //Crear el modelo y definir la cabecera de la tabla
        String titulos[] = {"DNI","Nombre","Apellidos","Sueldo","Fecha","Matricula"};
        m=new DefaultTableModel(null,titulos);

        String fila[] = new String[6];
        while(r.next()) {
            fila[0]=r.getString("DNI");
            fila[1]=r.getString("Nombre");
            fila[2]=r.getString("Apellidos");
            fila[3]=r.getString("Sueldo").replace(".",",");
            fecha=r.getString("Fecha");
            fecha=fecha.substring(8,10)+"/"+fecha.substring(5,7)+"/"+fecha.substring(0,4);
            fila[4]=fecha;
            fila[5]=r.getString("Matricula");
            m.addRow(fila);
        }
        tabla.setModel(m);

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Error al extraer los datos de la tabla");
    }
}
```

5. Lo que produce esta modificación del código es que al ejecutarse el programa se ejecute el método *MostrarTrabajadores* con lo que se ejecutará el código que hicimos en la hoja anterior para mostrar los datos de la tabla trabajadores en el JTable.

   Ejecuta el programa y comprueba el resultado.

6. Puesto que la tabla *trabajadores* se muestra al empezar el programa, la existencia del botón *Trabajadores* no tiene sentido, así pues elimina el botón *Trabajadores* de la ventana.

7. Añade ahora en la parte inferior un panel llamado *panelAcciones* y un botón *Eliminar Trabajador* llamado *btnEliminar*:

8. Se pretende que el usuario seleccione uno de los trabajadores de la tabla y al pulsar el botón *Eliminar* dicho trabajador se elimine de la base de datos. Esta eliminación por supuesto se verá reflejada en el JTable.

   Para ello, programe en el botón *Eliminar* lo siguiente:

```
    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int filsel; //fila seleccionada
        String dni;
        int resp; //respuesta

        try {
            filsel=tabla.getSelectedRow();
            if (filsel==-1) { //no se seleccionó ninguna fila
                JOptionPane.showMessageDialog(null,"Debes seleccionar el trabajador a borrar");
            } else { //sí se seleccionó una fila

                //pido confirmación
                resp=JOptionPane.showConfirmDialog(null,"¿Desea eliminar el trabajador seleccionado?",
                        "Eliminar",JOptionPane.YES_NO_OPTION);

                if (resp==JOptionPane.YES_OPTION) { //si quiero
                    m = (DefaultTableModel) tabla.getModel();
                    dni = (String) m.getValueAt(filsel,0); //extraigo el dni de la tabla
                    //elimino al que tenga ese deni
                    sentencia.executeUpdate("delete from trabajadores where dni='"+dni+"'");
                    MostrarTrabajadores(); //muestro los trabajadores
                }
            }
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al eliminar el trabajador");
        }
    }
```

Estudiemos con detenimiento el código programado.

Lo primero que se hace es recoger la fila seleccionada de la tabla usando el método *getSelectedRow*. Si el valor devuelto es –1 entonces es que no hay ninguna fila seleccionada. El programa avisa de esta circunstancia.

Si se seleccionó a un trabajador, entonces podemos borrar. Pero antes, es interesante pedir confirmación. Esto es lo que se hace con el JOptionPane.showConfirmDialog.

Si el usuario acepta la eliminación del trabajador, entonces la llevamos a cabo. Observa el proceso:

- Extraemos del modelo del JTable el dni del trabajador seleccionado (este dni se encuentra en la fila seleccionada –filsel- columna 0 –la primera columna-):

```
m = (DefaultTableModel) tabla.getModel();
dni = (String) m.getValueAt(filsel,0);
```

- Ahora se construirá una instrucción de acción SQL del tipo DELETE para que se elimine el trabajador con el dni extraído. Esto se hace concatenando y ejecutando la instrucción SQL a través del objeto *sentencia*:

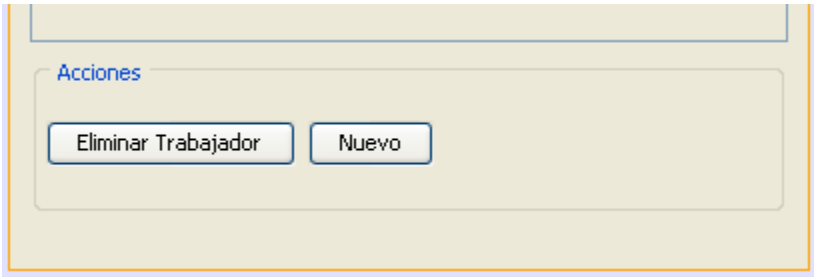
```
sentencia.executeUpdate("delete from trabajadores where dni='"+dni+"'");
```

- Y finalmente se llama al procedimiento *MostrarTabla* para que se extraiga de la base de datos y muestre de nuevo la tabla *trabajadores*. Esto actualizará el JTable, y se podrá ver que el trabajador ha sido eliminado.

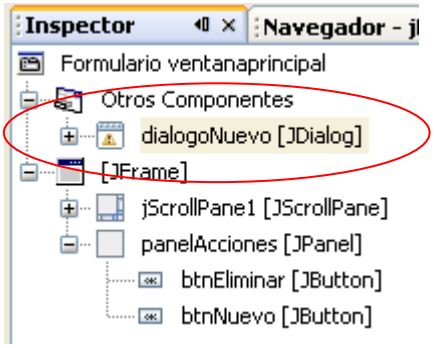
- Todo esto está dentro de un try...catch para capturar errores inesperados.

9. Ejecuta el programa y prueba a eliminar algún trabajador. Observa como el JTable se actualiza cuando se produce la eliminación.

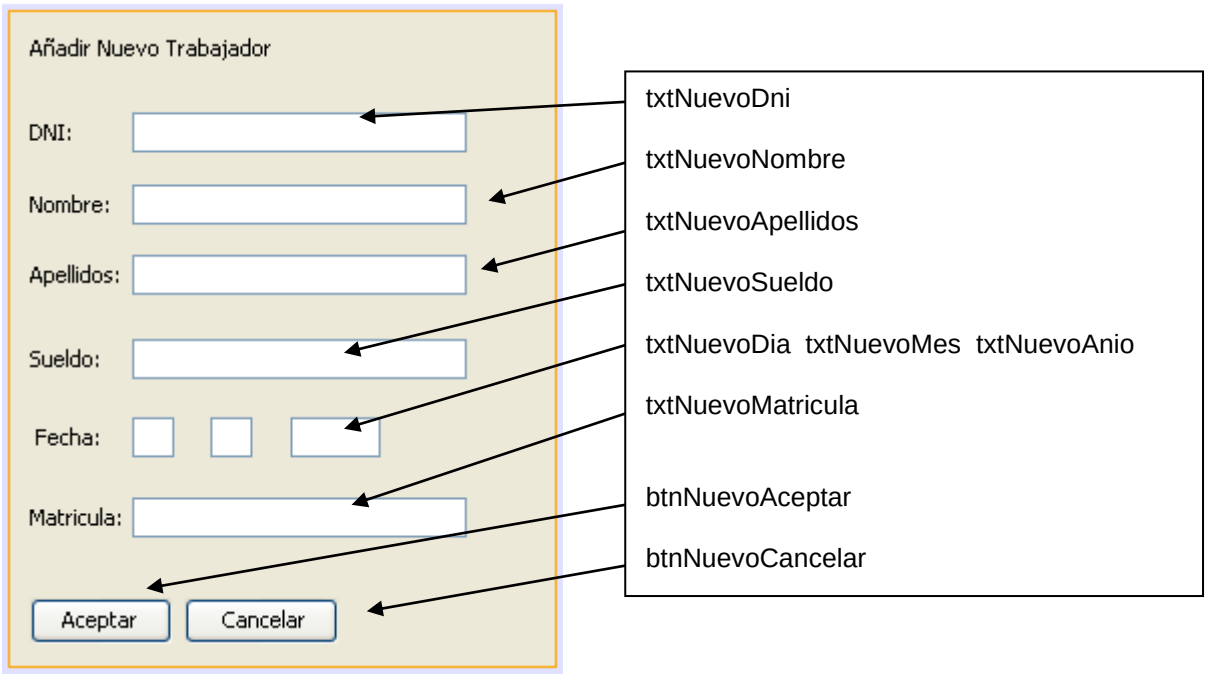
10. Ahora añade otro botón llamado *btnNuevo*:



11. Cuando se pulse el botón *Nuevo* se pretende que aparezca un formulario donde se puedan introducir los datos de un nuevo trabajador. Esto se conseguirá añadiendo un cuadro de diálogo a nuestro proyecto. Para ello, agrega un JDialog al proyecto. Este diálogo se llamará *dialogoNuevo*.



12. Haga doble clic en el *Inspector* sobre el *dialogoNuevo* para diseñarlo. Debe quedar como sigue, con los nombres que se indican a continuación:



13. La idea es la siguiente. Cuando el usuario pulse el botón *Nuevo*, aparecerá este cuadro de diálogo. El usuario introducirá los datos del nuevo trabajador y pulsará Aceptar, y entonces estos datos se introducirán en la tabla *trabajadores*.

    Si el usuario pulsa Cancelar, entonces no se hará nada.

    El JTable se actualizará para mostrar el resultado de la inserción del nuevo trabajador.

    Así pues, entre en el *actionPerformed* del botón *btnNuevo* y programe lo siguiente:

```
    private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        dialogoNuevo.setSize(300,400);
        dialogoNuevo.setModal(true);
        dialogoNuevo.setVisible(true);
        MostrarTrabajadores();
    }
```

    Este código empieza asignando un tamaño al cuadro de diálogo *dialogoNuevo*.

    Luego, se define el *dialogoNuevo* como Modal. Esto significa que hasta que no se termine de trabajar con este cuadro de diálogo no se podrá continuar usando el programa principal.

    Luego se muestra dicho cuadro de diálogo.

    Y finalmente se actualiza el JTable por si se hubiera introducido un nuevo trabajador.

14. Ahora programemos los botones del cuadro de diálogo *dialogoNuevo*. Empecemos por el botón *Cancelar*:

```
    private void btnNuevoCancelarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        dialogoNuevo.dispose();
    }
```

Como ves, tan sencillo como descargar el cuadro de diálogo. El botón *Cancelar* debe limitarse a quitar de la pantalla el cuadro de diálogo *dialogoNuevo*.

15. Y finalmente se programará el botón *Aceptar* del cuadro de diálogo. Recuerda que este botón es el que introduce en la base de datos a un nuevo trabajador. Programa dentro de este botón lo siguiente:

```
    private void btnNuevoAceptarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String dni,nombre,apell,sueldo,fecha,mat; //variables para datos
        String sql;

        try {
            //recojo los datos de los cuadros de texto
            dni=txtNuevoDni.getText();
            nombre=txtNuevoNombre.getText();
            apell=txtNuevoApellidos.getText();
            sueldo=txtNuevoSueldo.getText().replace(",",".");
            fecha=txtNuevoMes.getText()+"/"+txtNuevoDia.getText()+"/"+txtNuevoAnio.getText();
            mat=txtNuevoMatricula.getText();
            //construyo y ejecuto la instruccion sql
            sql="insert into trabajadores values('"+dni+"','"+nombre+"','"+apell+"',"+sueldo+",'"+
                    "#"+fecha+"#,'"+mat+"')";
            sentencia.executeUpdate(sql);
            dialogoNuevo.dispose();
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al introducir el nuevo trabajador");
        }
    }
```

Vamos a analizar detenidamente este código.

Lo primero que puedes observar es un conjunto de líneas que copian el contenido de los cuadros de texto en variables de cadena. Al hacer esto, cambiamos la coma decimal por punto decimal en el caso del sueldo y configuramos la fecha en el formato que entiende SQL: mes/dia/año.

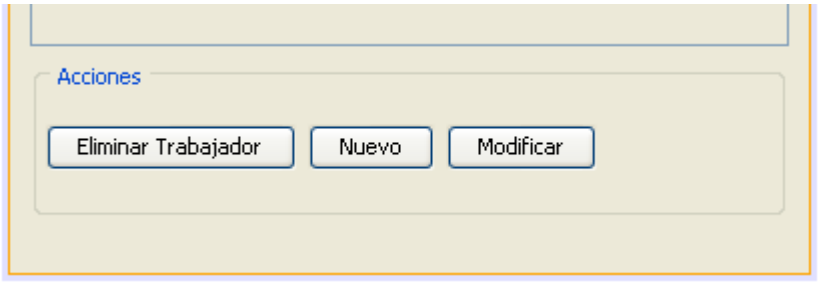
Luego, con estas variables, se construye por concatenación una instrucción SQL de tipo INSERT que permita introducir los datos del nuevo trabajador en la tabla *trabajadores*.

Se ejecuta dicha instrucción SQL. Y se cierra el cuadro de diálogo.

Por supuesto, es necesario un try...catch para evitar problemas inesperados.

16. Ejecuta el programa y prueba a introducir nuevos trabajadores.

17. Nuestro programa ya puede hacer altas y bajas. Solo queda que pueda realizar modificaciones. Para ello añade un nuevo botón a la ventana, llamado *btnModificar*:

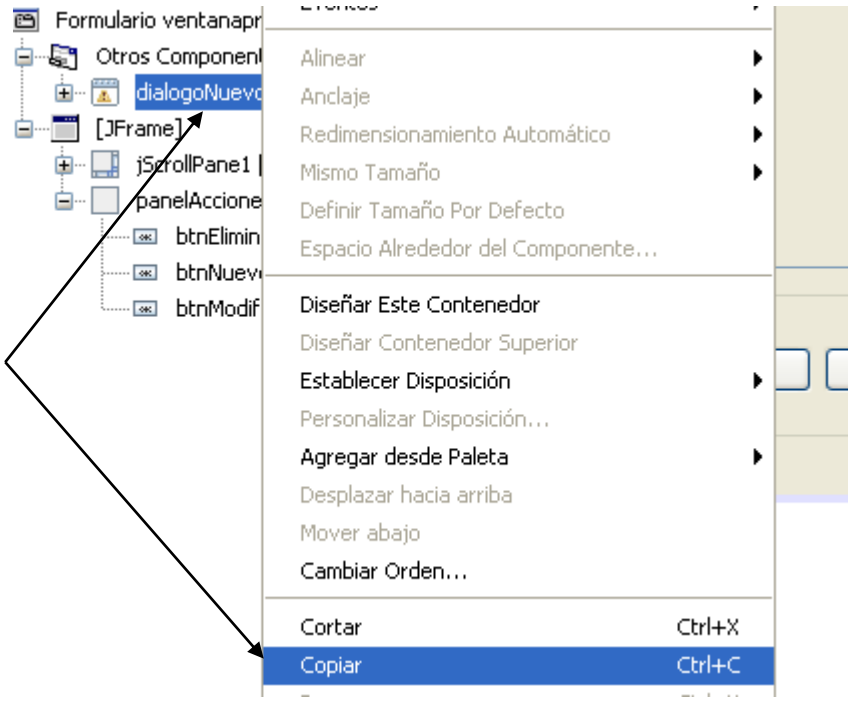


18. Se pretende que el usuario seleccione en la tabla el trabajador cuyos datos quiere modificar, y luego pulse este botón para efectuar la modificación.

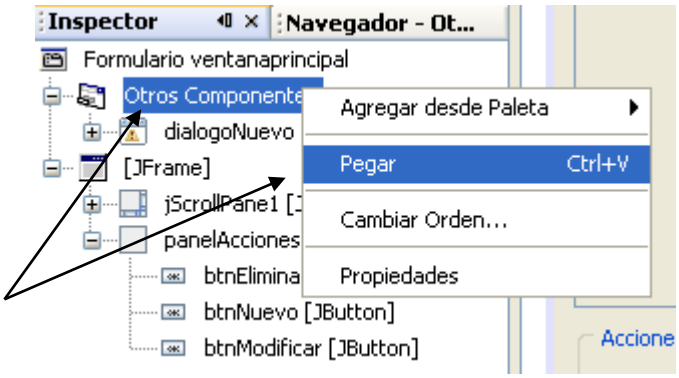
    Al pulsar este botón debe aparecer un cuadro de diálogo donde el usuario pueda cambiar fácilmente los datos.

    Ese cuadro de diálogo será muy parecido al que hemos hecho antes, así que básicamente solo tendrás que hacer una copia de dicho cuadro de diálogo y modificarlo un poco. A continuación se explica como hacerlo.

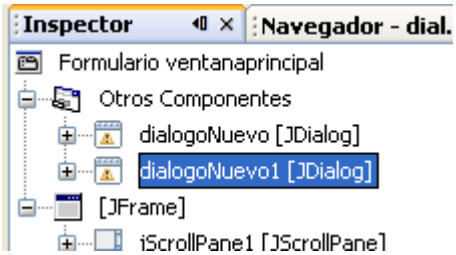
19. Haz clic con el derecho sobre el cuadro de diálogo *dialogoNuevo* y activa *Copiar* :



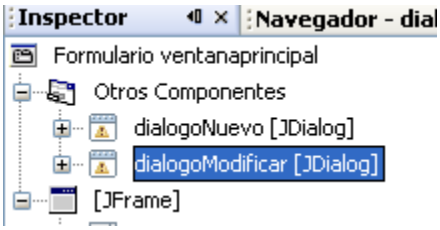
20. Luego activa *Pegar* sobre *Otros Componentes*:



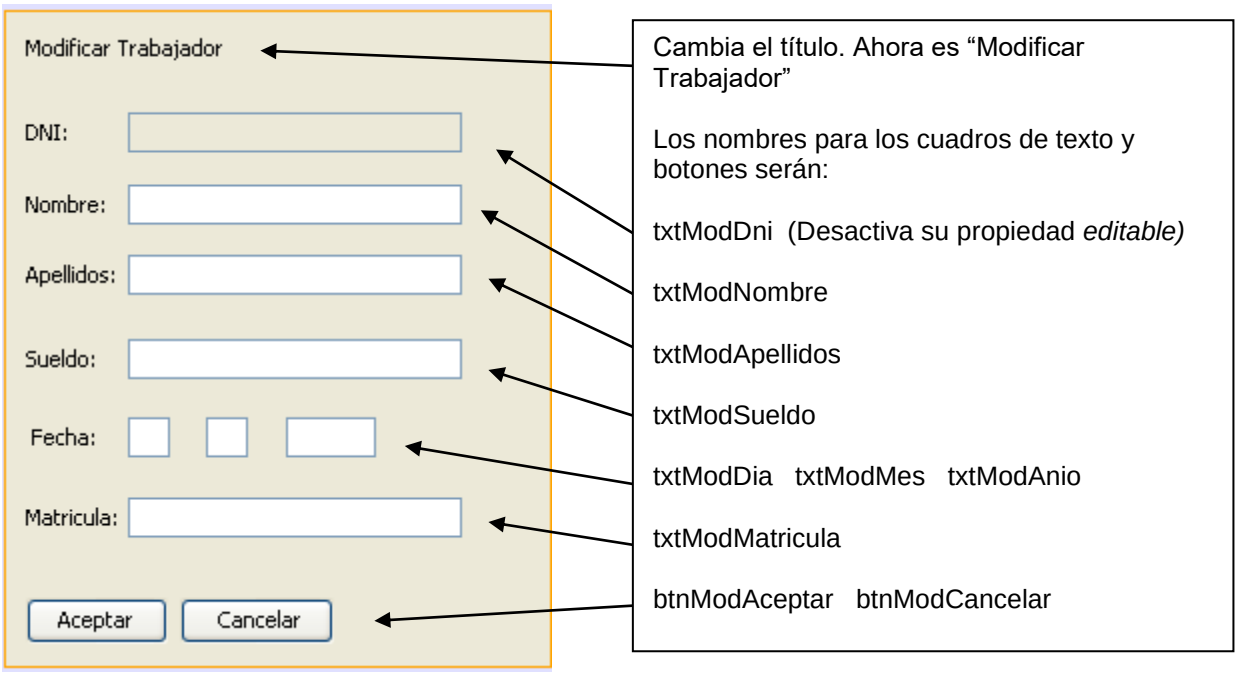
21. Aparecerá un nuevo cuadro de diálogo que es una copia del anterior. Este cuadro tendrá como nombre *dialogoNuevo1*.



22. Sin embargo, le cambiaremos el nombre para que sea más acorde con su función. Le llamaremos *dialogoModificar*. Cámbiale el nombre:



23. Vamos a modificar un poco el diseño del *dialogoModificar*. Haz doble clic sobre él y realiza las siguientes modificaciones en el diseño:



24. Empezaremos programando el botón *Modificar*. Al pulsar este botón se debe mostrar el cuadro de diálogo anterior relleno con los datos del trabajador que se quiere modificar. Se supone que el usuario ha seleccionado a este trabajador en la tabla anteriormente.

    Entra en el *actionPerformed* del botón *Modificar* y programa lo siguiente:

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    int fsel; //fila seleccionada
    String dni,nombre,apell,sueldo,fecha,mat; //cadenas de datos
    fsel=tabla.getSelectedRow();
    if (fsel==-1) { //no se seleccionó
        JOptionPane.showMessageDialog(null,"Debes seleccionar el trabajador a modificar");
    } else { //sí se seleccionó
        m = (DefaultTableModel)tabla.getModel();
        dni=(String)m.getValueAt(fsel,0);//extraigo los datos del JTable y los meto en variables
        nombre=(String)m.getValueAt(fsel,1);
        apell=(String)m.getValueAt(fsel,2);
        sueldo=(String)m.getValueAt(fsel,3);
        fecha=(String)m.getValueAt(fsel,4);
        mat=(String)m.getValueAt(fsel,5);
        txtModDni.setText(dni); //ahora pongo los datos extraidos en los cuadros de texto
        txtModNombre.setText(nombre);
        txtModApellidos.setText(apell);
        txtModSueldo.setText(sueldo);
        txtModDia.setText(fecha.substring(0,2));
        txtModMes.setText(fecha.substring(3,5));
        txtModAnio.setText(fecha.substring(6,10));
        txtModMatricula.setText(mat);
        //y finalmente se muestra el cuadro de diálogo
        dialogoModificar.setSize(300,400);
        dialogoModificar.setModal(true);
        dialogoModificar.setVisible(true);
        MostrarTrabajadores();
    }
}
```

Estudiemos el código.

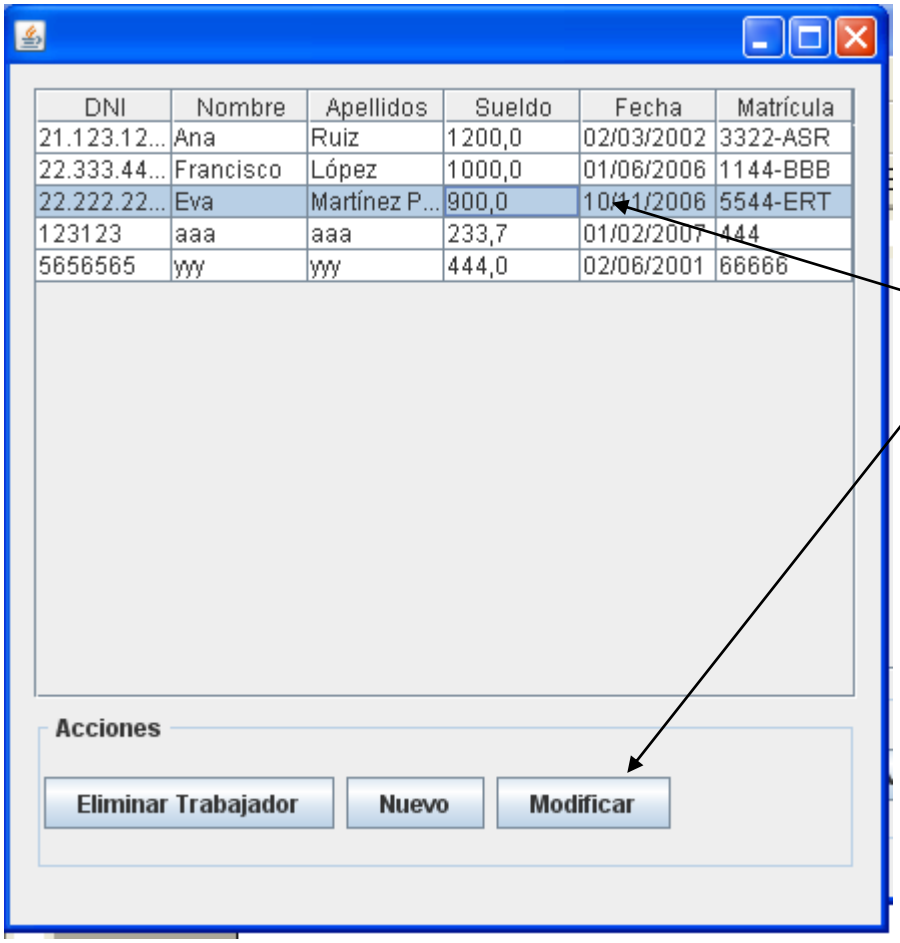
Primero se comprueba el número de la fila seleccionada. Si no hubiera ninguna se muestra un mensaje de error, ya que es necesario modificar la fila del trabajador que se quiere modificar.

En el caso de que haya una fila seleccionada, se extraen los datos del modelo del JTable y se almacenan en varias variables de cadena.

Una vez hecho esto, esas mismas variables se almacenan en los cuadros de texto del cuadro de diálogo *dialogoModificar*.

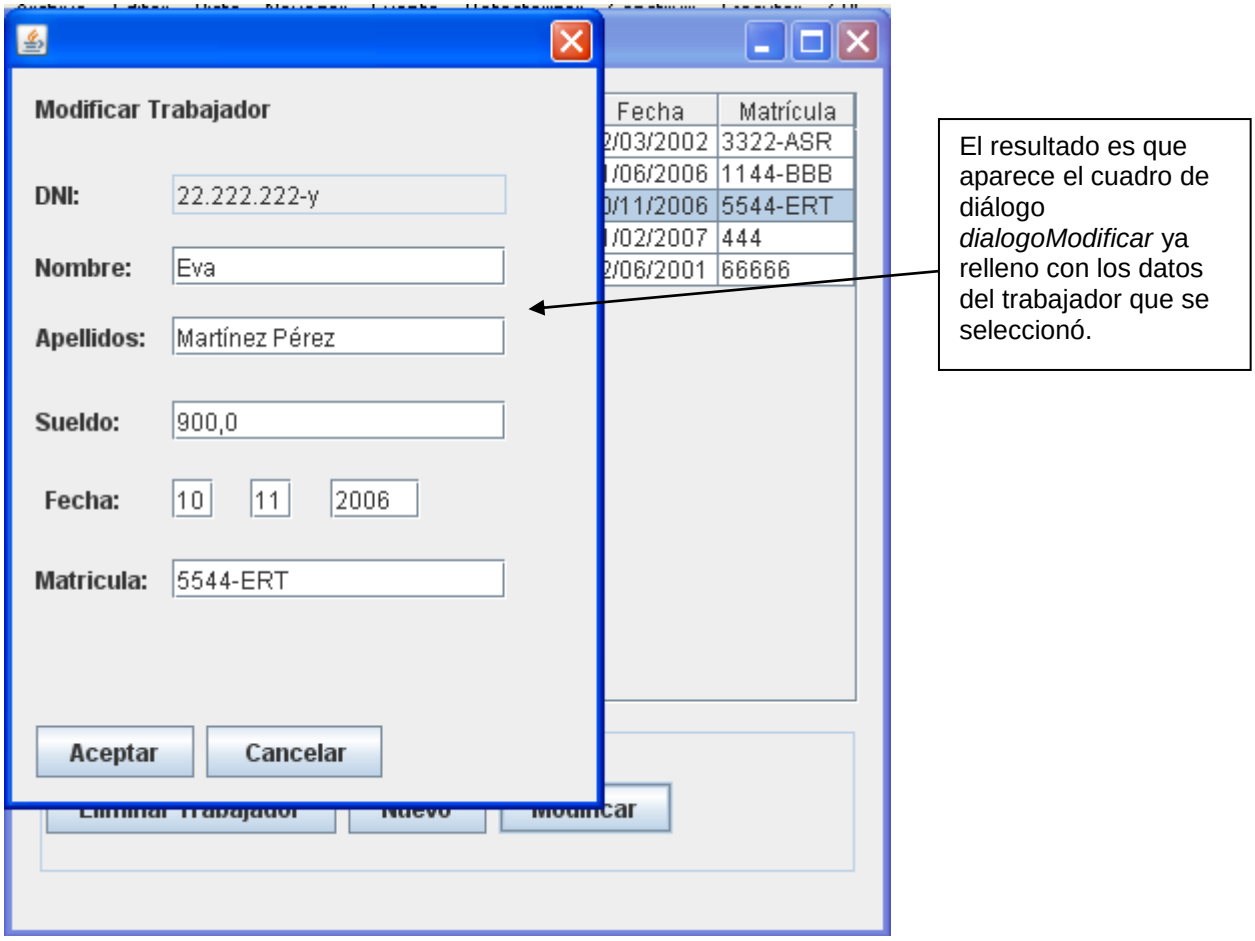
Y finalmente se prepara el cuadro de diálogo *dialogoModificar* y se muestra en la pantalla. Una vez realizada la modificación (no programada aún) se muestran los trabajadores en la tabla llamando al método *MostrarTrabajadores*.

25. Puedes comprobar el funcionamiento del programa de momento. Prueba a seleccionar una fila de la tabla y al pulsar el botón *Modificar.*



Selecciona...

Y luego pulsa modificar.



26. Lo bueno que tiene el rellenar el cuadro de diálogo *dialogoModificar* con los datos del trabajador que se quiere modificar es que no tenemos que escribir todos los datos, y solo modificar el campo que nos interese.

    El usuario realizará los cambios en los cuadros de textos ya rellenos y luego pulsará el botón *Aceptar* para que se produzca la modificación.

    Si se pulsa *Cancelar* no sucede nada. Simplemente se cerrará el cuadro de diálogo.

27. Se empezará programando el botón *Cancelar*. Este botón debe limitarse a cerrar el cuadro de diálogo *dialogoModificar*:

```
    private void btnModCancelarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        dialogoModificar.dispose();
    }
```

28. Ahora nos centraremos en el botón *Aceptar*. Este botón debe realizar la modificación, introduciendo todos los datos de los cuadros de texto en la tabla trabajadores.

    El código siguiente tomará como referencia el DNI del trabajador que se está modificando. Es importante que este DNI no cambie, ya que entonces estaríamos modificando los datos de otro trabajador.

Esta es la razón por la que el cuadro de texto *txtModDni* se ha configurado como *no editable*, para evitar que el usuario pueda cambiarlo accidentalente.

Programa el siguiente código en el botón *Aceptar*:

```
    private void btnModAceptarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String dni,nombre,apell,sueldo,fecha,mat; //variables para datos
        String sql;

        try {
            //recojo los datos de los cuadros de texto
            dni=txtModDni.getText();
            nombre=txtModNombre.getText();
            apell=txtModApellidos.getText();
            sueldo=txtModSueldo.getText().replace(",",".");
            fecha=txtModMes.getText()+"/"+txtModDia.getText()+"/"+txtModAnio.getText();
            mat=txtModMatricula.getText();
            //construyo y ejecuto la instruccion sql
            sql="update trabajadores set "+
                    "nombre='"+nombre+"',"+
                    "apellidos='"+apell+"',"+
                    "sueldo="+sueldo+","+
                    "fecha=#"+fecha+"#,"+
                    "matricula='"+mat+"' "+
                    "where dni='"+dni+"'";
            sentencia.executeUpdate(sql);
            dialogoModificar.dispose();
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al modificar el trabajador");
        }
    }
```

Estudiemos el código.

Lo primero que se hace es recoger en variables los datos introducidos en los cuadros de texto.

Luego, con estas variables, se construye una instrucción SQL del tipo UPDATE que permite modificar los datos del trabajador con el dni indicado en el cuadro de diálogo.

Finalmente se cierra el cuadro de diálogo.

Todo este código es susceptible de sufrir fallos por lo que está rodeado de un try...catch.

29. Ejecuta el programa y comprueba el funcionamiento de la actualización de trabajadores. Prueba a realizar varias actualizaciones. Observa como el JTable se actualiza con las nuevas modificaciones realizadas.

CONCLUSIÓN

La más simple de las aplicaciones de base de datos debe ser capaz de realizar las siguientes operaciones sobre una tabla:

- **Altas**
- **Bajas**
- **Modificaciones**

Es muy interesante que la aplicación muestre en pantalla continuamente un JTable con el contenido de la tabla de la base de datos sobre la que se está trabajando.

El JTable nos permite visualizar los datos de la tabla y seleccionar rápidamente el registro que queramos manipular.

Para la inserción de nuevos registros en la tabla se recomienda la creación de un cuadro de diálogo que muestre un formulario donde el usuario pueda introducir los datos del nuevo registro cómodamente.

Para la modificación de un registro, se recomienda la creación de otro cuadro de diálogo que muestre los datos del registro que se quiere modificar.