

10/6/2019



Práctica: Modelo Vista Controlador

Entornos de desarrollo / Programación



Andrés Ceballos Rodríguez

Práctica: Modelo Vista Controlador

Base de datos

Índice

1. Introducción	2
2. Clase Principal.....	2
3. Clase Modelo.....	2
4. Controlador	5
5. Clase Menú Principal.....	14
6. Demandantes: Baja	15
7. Ofertas: Modificación y consulta.....	17
8. Gestión: Alta	21
9. Bibliografía.....	24

1. Introducción

En esta práctica vamos a crear un pequeño programa en Java conectado a una Base de datos MySQL, realizado siguiendo el método Modelo Vista Controlador.

En este método, se divide por un lado la interfaz gráfica, los métodos que se emplean y el controlador que es el encargado de unir la vista (interfaz gráfica) y los diferentes métodos (Modelo).

2. Clase Principal

En primer lugar, vamos a pasar a mostrar el Menú principal. Para cargar esta clase y todo el programa necesitaremos una clase principal con su main correspondiente que se encargará de ejecutar el controlador en el que incluiremos por parámetros el menú principal y el Modelo.

Aquí tenemos el código de la clase Principal.java que hemos descrito anteriormente.

```
package es.studium.PracticaMVC;

public class Principal {

    public static void main(String[] args) {
        Modelo Model = new Modelo();
        MenuPrincipal MenuPrinci = new MenuPrincipal();
        new Controlador(Model, MenuPrinci);
    }
}
```

3. Clase Modelo

En el modelo encontramos los diferentes métodos que vamos a usar para el programa.

En nuestro caso, hemos introducido métodos para realizar conexiones y desconexiones a la base de datos y consultas de tipo SELECT, INSERT Y DELETE gracias a el paquete de java: SQL.

También encontramos en el modelo un método de calendario que nos hace pasar las fechas de formato Europeo a Formato Americano.

Código Modelo:

```
package es.studium.PracticaMVC;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Calendar;

import javax.swing.JOptionPane;

public class Modelo
{
    public Connection conectar(String baseDatos, String usuario,
String clave)
    {
        String driver = "com.mysql.jdbc.Driver";
        String url
="jdbc:mysql://localhost:3306/" + baseDatos + "?autoReconnect=true&useSSL=false";
        String login = usuario;
        String password = clave;
        Connection connection = null;

        try
        {
            Class.forName(driver);
            connection = DriverManager.getConnection(url,
login,password);
        }
        catch (ClassNotFoundException cnfe)
        {
            JOptionPane.showMessageDialog(null, cnfe.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
        catch (SQLException sqle)
        {
            JOptionPane.showMessageDialog(null, sqle.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
        return connection;
    }
    public void desconectar(Connection c)
```

```
{
    try
    {
        if(c!=null)
        {
            c.close();
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
    }
}
public ResultSet ejecutarSelect(String sentencia, Connection c)
{
    try
    {
        Statement statement = c.createStatement();
        ResultSet rs= statement.executeQuery(sentencia);
        return rs;
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
        return null;
    }
}
public void ejecutarIDA(String sentencia, Connection c)
{
    try
    {
        Statement statement = c.createStatement();
        statement.executeUpdate(sentencia);
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
    }
}
```

```
}  
public String Calendario() {  
    Calendar horaFecha = Calendar.getInstance();  
    int dia,mes,anyo;  
  
    dia = horaFecha.get(Calendar.DAY_OF_MONTH);  
    mes = horaFecha.get(Calendar.MONTH)+1;  
    anyo = horaFecha.get(Calendar.YEAR);  
  
    if(dia<10 && mes<10)  
    {  
        return "0"+dia+"/"+"0"+mes+"/"+anyo;  
    } else if(dia<10 && mes>=10)  
    {  
        return "0"+dia+"/"+mes+"/"+anyo;  
    } else if (dia>=10 && mes<10)  
    {  
        return dia+"/"+"0"+mes+"/"+anyo;  
    } else if(dia>=10 && mes<=10)  
    {  
        return dia+"/"+mes+"/"+anyo;  
    }  
    return dia+"/"+mes+"/"+anyo;  
}  
}
```

4. Controlador

En el controlador encontramos todo el proceso que une la vista como el modelo.

Aquí vamos a ver todas las interacciones con los métodos de la clase Modelo, así como las interacciones con los distintos botones, cuadros de texto y diálogos.

Código del Controlador:

```
package es.studium.PracticaMVC;  
  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;
```

```

import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class Controlador implements ActionListener, WindowListener
{
    Modelo Model;
    MenuPrincipal MenuPrinci;
    BajaDemandante BajaDem;
    ModificacionOferta ModOf;
    EdicionOferta EdOf;
    AltaAsignacion Aa;
    ConsultaOfertas ConOf;
    int ofertaSeleccionada = 0;
    int demandanteSeleccionado = 0;
    public Controlador(Modelo m, MenuPrincipal Mp) {
        Model = m;
        MenuPrinci = Mp;
        Mp.mniDemandantesBaja.addActionListener(this);
        Mp.mniOfertasModificacion.addActionListener(this);
        Mp.mniOfertasConsulta.addActionListener(this);
        Mp.mniGestionAlta.addActionListener(this);
        Mp.addWindowListener(this);
    }
    @Override
    public void actionPerformed(ActionEvent ae)
    {
        //BAJA DEMANDANTE
        if(MenuPrinci.mniDemandantesBaja.equals(ae.getSource()))
        {
            BajaDem = new BajaDemandante();
            MenuPrinci.setVisible(false);
            BajaDem.addWindowListener(this);
            BajaDem.btnEliminar.addActionListener(this);
            BajaDem.btnCancelar.addActionListener(this);
            ResultSet dm = Model.ejecutarSelect("SELECT * FROM
demandantes", Model.conectar("practicamvc","root","Studium2018;"));
            try {
                while(dm.next())
                {
                    String
demandantes=Integer.toString(dm.getInt("idDemandante"));
                    demandantes = demandantes+".-"+
dm.getString("nombreDemandante")+" "+
dm.getString("apellidosDemandante")+" -
"+dm.getString("dniDemandante");
                    BajaDem.demandantes.add(demandantes);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    }
    } catch (SQLException e) {

        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
    }

    Model.desconectar(Model.conectar("practicamvc","root"
    ,"Studium2018;"));
    }
    try {
        if(BajaDem.btnEliminar.equals(ae.getSource()))
        {
            int seleccion = JOptionPane.showOptionDialog(
            null,"¿Desea eliminar demandante?", "Eliminar
            demandante",JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.QUESTION_ME
            SSAGE,null,new Object[] { "Eliminar", "Cancelar"},"Cancelar");

            if (seleccion == 0)
            {
                try
                {
                    String[] array=
                    BajaDem.demandantes.getSelectedItem().toString().split("-");
                    demandanteSeleccionado =
                    Integer.parseInt(array[0]);
                } catch (NumberFormatException Nf)
                {

                    JOptionPane.showMessageDialog(null,"Introduzca demandante
                    válido","Error de demandante", JOptionPane.ERROR_MESSAGE);
                }
                Model.ejecutarIDA("DELETE FROM
                demandantes where idDemandante =" +demandanteSeleccionado+";",
                Model.conectar("practicamvc", "root", "Studium2018;"));

                JOptionPane.showMessageDialog(null,"Demandante eliminado con
                éxito","Demandante eliminado", JOptionPane.INFORMATION_MESSAGE);
            } else if(seleccion == 1){}
        }
        if(BajaDem.btnCancelar.equals(ae.getSource())) {
            BajaDem.setVisible(false);
            MenuPrinci.setVisible(true);
        }
    } catch (NullPointerException np) {}
    //MODIFICAR OFERTA

```



```

        if(MenuPrinci.mniOfertasModificacion.equals(ae.getSource()))
        {
            ModOf = new ModificacionOferta();
            MenuPrinci.setVisible(false);
            ModOf.addWindowListener(this);
            ModOf.btnEditar.addActionListener(this);
            ModOf.btnCancel.addActionListener(this);
            ResultSet of = Model.ejecutarSelect("SELECT
idOferta,DATE_FORMAT(fechaOferta,
'%d/%m/%Y'),DATE_FORMAT(fechaFinOferta, '%d/%m/%Y') FROM ofertas",
Model.conectar("practicamvc","root","Stodium2018;"));
            try
            {
                while(of.next())
                {
                    String
ofertas=Integer.toString(of.getInt("idOferta"));
                    ofertas = ofertas+"-"+ "Fecha
Oferta:"+of.getString("DATE_FORMAT(fechaOferta, '%d/%m/%Y')")+ "
Fecha Fin Oferta:"+ of.getString("DATE_FORMAT(fechaFinOferta,
'%d/%m/%Y')");
                    ModOf.ofertas.add(ofertas);
                }
            } catch (SQLException e) {

                JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
            }

            Model.desconectar(Model.conectar("practicamvc","root"
,"Stodium2018;"));
        }
        //EDICIÓN OFERTA
        try {
            if(ModOf.btnEditar.equals(ae.getSource()))
            {
                String[] array=
ModOf.ofertas.getSelectedItemAt(0).toString().split("-");
                ofertaSeleccionada =
Integer.parseInt(array[0]);
                EdOf = new EdicionOferta(ofertaSeleccionada);
                EdOf.btnActualizar.addActionListener(this);
                EdOf.btnCancel.addActionListener(this);
                EdOf.addWindowListener(this);
                ModOf.setVisible(false);
                ResultSet ofSel = Model.ejecutarSelect("SELECT
DATE_FORMAT(fechaOferta, '%d/%m/%Y'),DATE_FORMAT(fechaFinOferta,

```

```

'%d/%m/%Y'),requisitosOferta FROM ofertas where idOferta
="+ofertaSeleccionada+";",
Model.conectar("practicamvc","root","Stodium2018;"));
        try
        {
            ofSel.next();

            EdOf.txtFecha.setText(ofSel.getString("DATE_FORMAT(fechaOferta,
'%d/%m/%Y')"));

            EdOf.txtFechaFin.setText(ofSel.getString("DATE_FORMAT(fechaFinO
ferta, '%d/%m/%Y')"));

            EdOf.txtRequisitos.setText(ofSel.getString("requisitosOferta"))
;

        } catch (SQLException e)
        {

            JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
        }

        Model.desconectar(Model.conectar("practicamvc","root"
,"Stodium2018;"));
        }
        else if(ModOf.btnCancelar.equals(ae.getSource()))
        {
            ModOf.setVisible(false);
            MenuPrinci.setVisible(true);
        }
        if(EdOf.btnActualizar.equals(ae.getSource()))
        {
            String Fecha = EdOf.txtFecha.getText();
            String[] arrayFecha = Fecha.split("/");
            String FechaFin = EdOf.txtFechaFin.getText();
            String[] arrayFechaFin = FechaFin.split("/");
            try
            {
                Fecha = arrayFecha[2]+"-
"+arrayFecha[1]+"-"+arrayFecha[0];
                FechaFin = arrayFechaFin[2]+"-
"+arrayFechaFin[1]+"-"+arrayFechaFin[0];
            } catch (ArrayIndexOutOfBoundsException ai) {}
            Model.ejecutarIDA("UPDATE ofertas SET
fechaOferta='"+Fecha+"', fechaFinOferta='"+FechaFin+"',
requisitosOferta='"+EdOf.txtRequisitos.getText()+"' WHERE idOferta

```

```

="+ofertaSeleccionada+";", Model.conectar("practicamvc","root"
,"Stodium2018;"));
        JOptionPane.showMessageDialog(null,"Oferta
Modificada con éxito","Oferta Modificada",
JOptionPane.INFORMATION_MESSAGE);
    }
    else if (EdOf.btnCancelar.equals(ae.getSource()))
    {
        EdOf.setVisible(false);
        MenuPrinci.setVisible(true);
    }
} catch (NullPointerException np) {}

//CONSULTA OFERTAS
try
{
    if(MenuPrinci.mniOfertasConsulta.equals(ae.getSource()))
    {
        ConOf = new ConsultaOfertas();
        MenuPrinci.setVisible(false);
        ConOf.addWindowListener(this);
        ConOf.btnAceptar.addActionListener(this);
        ConOf.modelo.addColumn("Número");
        ConOf.modelo.addColumn("Número Demandantes");
        ConOf.modelo.addColumn("Fecha Fin");
        ResultSet Co = Model.ejecutarSelect("select
idOfertaFK, count(idDemandanteFK), DATE_FORMAT(fechaFinOferta,
'%d/%m/%Y') from asignaciones join ofertas on idOfertaFK = idOferta
group by idOfertaFK order by idOfertaFK;",
Model.conectar("practicamvc","root" ,"Stodium2018;"));
        try {
            while (Co.next())
            {
                Object [] fila = new Object[3];
                for (int i=0;i<3;i++)
                    if(i==2) {
                        fila[i] =
Co.getString("DATE_FORMAT(fechaFinOferta, '%d/%m/%Y')");
                    } else {
                        fila[i] =
Co.getObject(i+1);
                    }
                ConOf.modelo.addRow(fila);
            }
        } catch (SQLException e)
        {

```

```

        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
    }

    Model.desconectar(Model.conectar("practicamvc","root"
    ,"Stodium2018;"));
        ConOf.tablaOfertas.setEnabled(false);
    }
    if(ConOf.btnAceptar.equals(ae.getSource()))
    {
        ConOf.setVisible(false);
        MenuPrinci.setVisible(true);
    }

} catch (NullPointerException np) {}
try
{
    if(MenuPrinci.mniGestionAlta.equals(ae.getSource()))
    {
        Aa = new AltaAsignacion();
        MenuPrinci.setVisible(false);
        Aa.btnAceptar.addActionListener(this);
        Aa.btnCancelar.addActionListener(this);
        Aa.addWindowListener(this);
        //CHOICE OFERTAS
        ResultSet Aof = Model.ejecutarSelect("SELECT
idOferta, DATE_FORMAT(fechaOferta,
'%d/%m/%Y'),DATE_FORMAT(fechaFinOferta, '%d/%m/%Y') FROM ofertas;",
Model.conectar("practicamvc","root","Stodium2018;"));
        try
        {
            while(Aof.next())
            {
                String
ofertas=Integer.toString(Aof.getInt("idOferta"));
                ofertas = ofertas+"-"+ "Fecha
Oferta:"+Aof.getString("DATE_FORMAT(fechaOferta, '%d/%m/%Y')")+
Fecha Fin Oferta:"+ Aof.getString("DATE_FORMAT(fechaFinOferta,
'%d/%m/%Y')");

                Aa.ofertas.add(ofertas);
            }
        } catch (SQLException e)
        {

        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
        JOptionPane.ERROR_MESSAGE);
    }
}

```

```

    }

    Model.desconectar(Model.conectar("practicamvc","root"
,"Stodium2018;"));

    //CHOICE DEMANDANTES
    ResultSet Adm = Model.ejecutarSelect("SELECT *
FROM demandantes",
Model.conectar("practicamvc","root","Stodium2018;"));
    try
    {
        while(Adm.next())
        {
            String
demandantes=Integer.toString(Adm.getInt("idDemandante"));
            demandantes = demandantes+".-"+
Adm.getString("nombreDemandante")+" "+
Adm.getString("apellidosDemandante")+" -
"+Adm.getString("dniDemandante");
            Aa.demandantes.add(demandantes);
        }
    } catch (SQLException e)
    {

        JOptionPane.showMessageDialog(null,e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
    }

    Model.desconectar(Model.conectar("practicamvc","root"
,"Stodium2018;"));
    Aa.txtFecha.setText(Model.Calendario());
}
if(Aa.btnAceptar.equals(ae.getSource()))
{
    String Fecha = Aa.txtFecha.getText();
    String[] arrayFecha = Fecha.split("/");
    try
    {
        Fecha = arrayFecha[2]+"-
"+arrayFecha[1]+"-"+arrayFecha[0];
    } catch (ArrayIndexOutOfBoundsException ai) {}
    String[] arrayOferta=
Aa.ofertas.getSelectedItemAt().toString().split("-");
    ofertaSeleccionada =
Integer.parseInt(arrayOferta[0]);
    String[] arrayDemanda=
Aa.demandantes.getSelectedItemAt().toString().split("-");

```

```

        demandanteSeleccionado =
Integer.parseInt(arrayDemanda[0]);
        Model.ejecutarIDA("INSERT INTO asignaciones
VALUES (null,'"+Fecha+"', '"+ofertaSeleccionada+"',
'"+demandanteSeleccionado+"');",Model.conectar("practicamvc","root",
"Stadium2018;"));
        JOptionPane.showMessageDialog(null,"Asignación
añadida con éxito","Asignación añadida",
JOptionPane.INFORMATION_MESSAGE);
    } else if (Aa.btnCancelar.equals(ae.getSource()))
    {
        MenuPrinci.setVisible(true);
        Aa.setVisible(false);
    }
    } catch (NullPointerException np){}
}
@Override
public void windowActivated(WindowEvent arg0) {}
@Override
public void windowClosed(WindowEvent arg0) {}
@Override
public void windowClosing(WindowEvent arg0)
{
    System.exit(0);
}
@Override
public void windowDeactivated(WindowEvent arg0) {}

@Override
public void windowDeiconified(WindowEvent arg0) {}

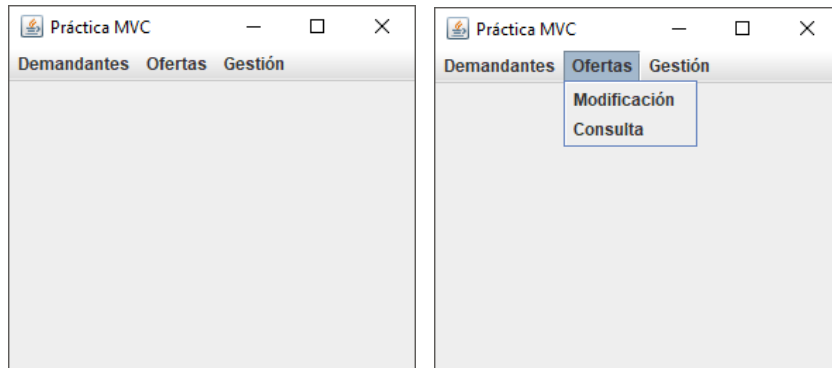
@Override
public void windowIconified(WindowEvent arg0) {}

@Override
public void windowOpened(WindowEvent arg0) {}
}

```

5. Clase Menú Principal

Ahora, pasamos al menú principal que es dónde encontramos la primera interfaz gráfica. En él tenemos todas las opciones para trabajar con el programa.



Código de la Clase Menú Principal:

```
package es.studium.PracticaMVC;

import java.awt.FlowLayout;
import javax.swing.*.*;

public class MenuPrincipal extends JFrame
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    JMenuBar barraMenu = new JMenuBar();
    JMenu menuDemandantes = new JMenu("Demandantes");
    JMenu menuOfertas = new JMenu("Ofertas");
    JMenu menuGestion = new JMenu("Gestión");
    JMenuItem mniDemandantesBaja = new JMenuItem("Baja");
    JMenuItem mniOfertasModificacion = new
JMenuItem("Modificación");
    JMenuItem mniOfertasConsulta = new JMenuItem("Consulta");
    JMenuItem mniGestionAlta = new JMenuItem("Alta");
    JPanel pnl1 = new JPanel();

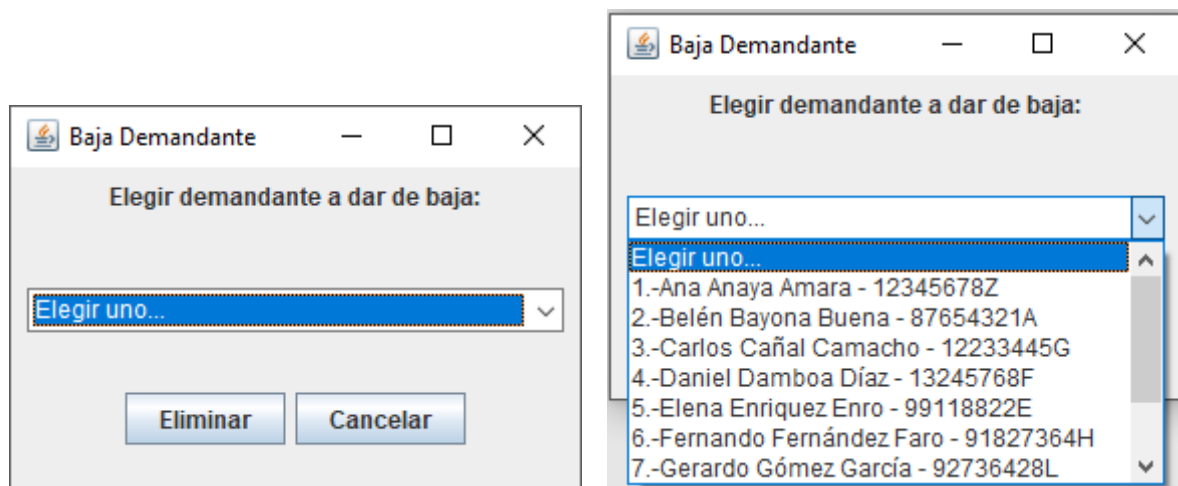
    public MenuPrincipal() {
```

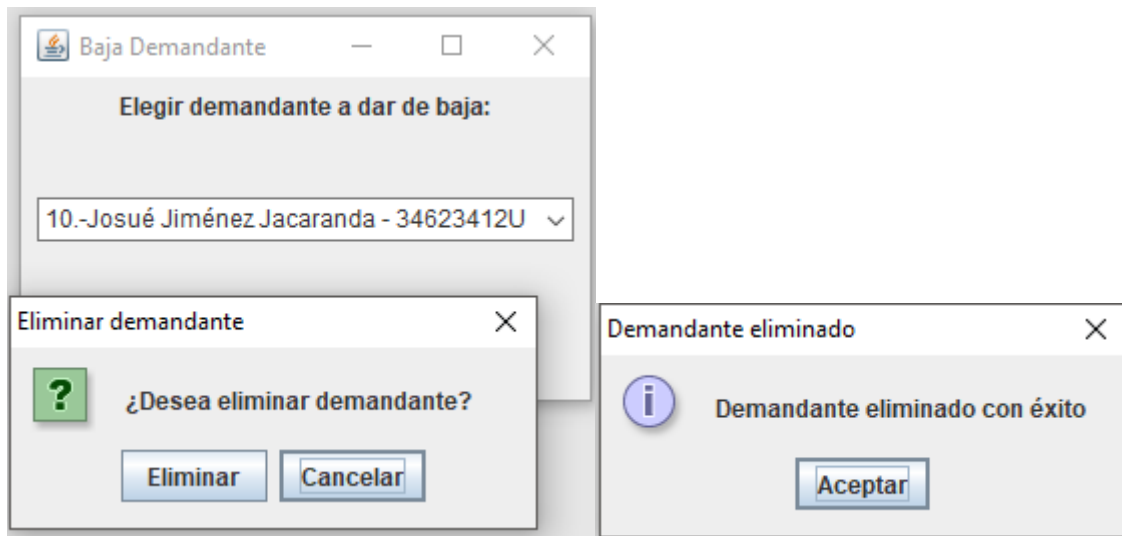
```
        this.setSize(300,250);
        this.setTitle("Práctica MVC");
        this.setLocationRelativeTo(null);
        this.setLayout(new FlowLayout());
        this.setJMenuBar(barraMenu);
        menuDemandantes.add(mniDemandantesBaja);
        menuOfertas.add(mniOfertasModificacion);
        menuOfertas.add(mniOfertasConsulta);
        menuGestion.add(mniGestionAlta);
        barraMenu.add(menuDemandantes);
        barraMenu.add(menuOfertas);
        barraMenu.add(menuGestion);
        this.add(pnl1);
        this.setVisible(true);
    }
}
```

6. Demandantes: Baja

La siguiente clase que nos encontramos es la Baja de demandante. Una clase en la que vamos a dar de baja a un demandante.

Primero elegimos el demandante y luego nos aparece un diálogo si queremos aceptar eliminarlo o no.





Código de BajaDemandante:

```
package es.studium.PracticaMVC;

import java.awt.Choice;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

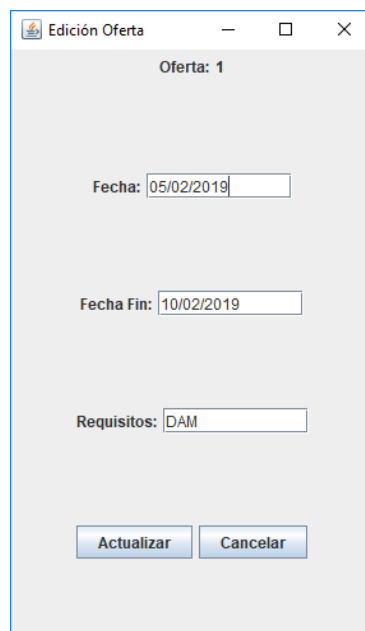
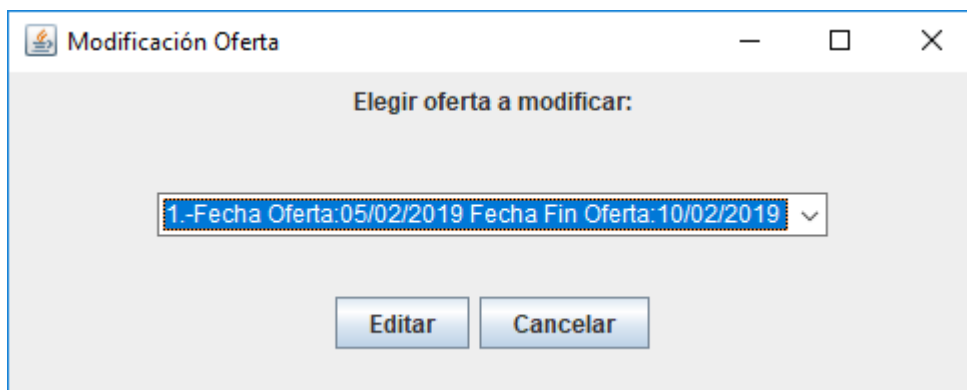
public class BajaDemandante extends JFrame
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JLabel lblElegir = new JLabel("Elegir demandante a dar de
baja:");
    Choice demandantes = new Choice();
    JButton btnEliminar = new JButton("Eliminar");
    JButton btnCancelar = new JButton("Cancelar");
    JPanel pnl1 = new JPanel();
    JPanel pnl2 = new JPanel();
    JPanel pnl3 = new JPanel();
    public BajaDemandante() {
        this.setSize(300,200);
        this.setTitle("Baja Demandante");
        this.setLocationRelativeTo(null);
        this.setLayout(new GridLayout(3,1));
        pnl1.add(lblElegir);
        demandantes.add("Elegir uno...");
```

```
        pnl2.add(demandantes);  
        pnl3.add(btnEliminar);  
        pnl3.add(btnCancelar);  
        this.add(pnl1);  
        this.add(pnl2);  
        this.add(pnl3);  
        this.setVisible(true);  
    }  
}
```

7. Ofertas: Modificación y consulta

En las ofertas encontramos tres clases, dos para la modificación de las ofertas y otra para la consulta de las ofertas.

En cuanto a la modificación, encontramos primero una clase para elegir la oferta a modificar y otra para modificar la oferta seleccionada.



Código Modificación Oferta:

```
package es.studium.PracticaMVC;

import java.awt.Choice;
import java.awt.GridLayout;
import javax.swing.*;

public class ModificacionOferta extends JFrame
{
    JLabel lblModOf = new JLabel("Elegir oferta a modificar:");
    Choice ofertas = new Choice();
    JButton btnEditar = new JButton("Editar");
    JButton btnCancelar = new JButton("Cancelar");
    JPanel pnl1 = new JPanel();
    JPanel pnl2 = new JPanel();
    JPanel pnl3 = new JPanel();
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public ModificacionOferta() {
        this.setTitle("Modificación Oferta");
        this.setLayout(new GridLayout (3,1));
        this.setSize(500, 200);
        this.setLocationRelativeTo(null);
        pnl1.add(lblModOf);
        pnl2.add(ofertas);
        pnl3.add(btnEditar);
        pnl3.add(btnCancelar);
        this.add(pnl1);
        this.add(pnl2);
        this.add(pnl3);
        this.setVisible(true);
    }
}
```

Código EdicionOferta:

```
package es.studium.PracticaMVC;

import java.awt.GridLayout;
import javax.swing.*.*;

public class EdicionOferta extends JFrame {

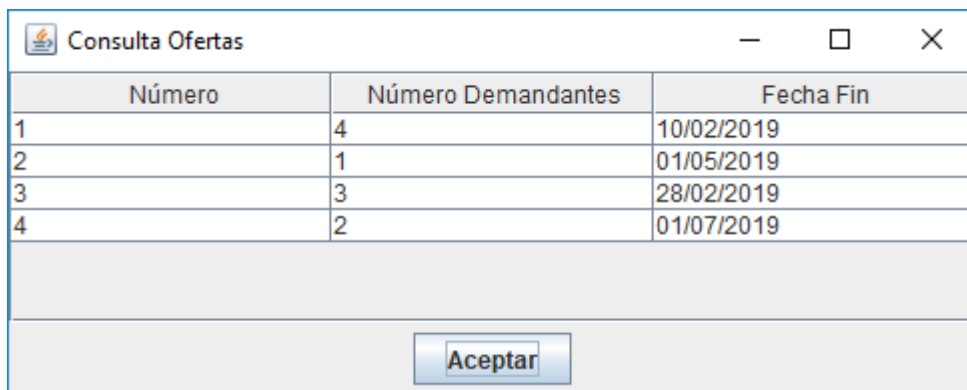
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    String idOf;
    JLabel lblOferta = new JLabel("Oferta:");
    JLabel lblNumOferta = new JLabel("");
    JLabel lblFecha = new JLabel("Fecha:");
    JLabel lblFechaFin = new JLabel("Fecha Fin:");
    JLabel lblRequisitos = new JLabel("Requisitos:");
    JTextField txtFecha = new JTextField(10);
    JTextField txtFechaFin = new JTextField(10);
    JTextField txtRequisitos = new JTextField(10);
    JButton btnActualizar = new JButton("Actualizar");
    JButton btnCancelar = new JButton("Cancelar");
    JPanel pnl1 = new JPanel();
    JPanel pnl2 = new JPanel();
    JPanel pnl3 = new JPanel();
    JPanel pnl4 = new JPanel();
    JPanel pnl5 = new JPanel();
    public EdicionOferta(int id) {
        this.setTitle("Edición Oferta");
        this.setLayout(new GridLayout(5,1));
        this.setSize(300, 500);
        this.setLocationRelativeTo(null);
        idOf = Integer.toString(id);
        lblNumOferta.setText(idOf);
        pnl1.add(lblOferta);
        pnl1.add(lblNumOferta);
        pnl2.add(lblFecha);
        pnl2.add(txtFecha);
        pnl3.add(lblFechaFin);
        pnl3.add(txtFechaFin);
        pnl4.add(lblRequisitos);
        pnl4.add(txtRequisitos);
        pnl5.add(btnActualizar);
        pnl5.add(btnCancelar);
        this.add(pnl1);
```

```

        this.add(pnl2);
        this.add(pnl3);
        this.add(pnl4);
        this.add(pnl5);
        this.setVisible(true);
    }
}

```

Para la consulta, encontramos una sola clase en la que se nos muestra una consulta en forma de Tabla:



Número	Número Demandantes	Fecha Fin
1	4	10/02/2019
2	1	01/05/2019
3	3	28/02/2019
4	2	01/07/2019

Aceptar

Código ConsultaOfertas:

```

package es.studium.PracticaMVC;

import java.awt.BorderLayout;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class ConsultaOfertas extends JFrame {

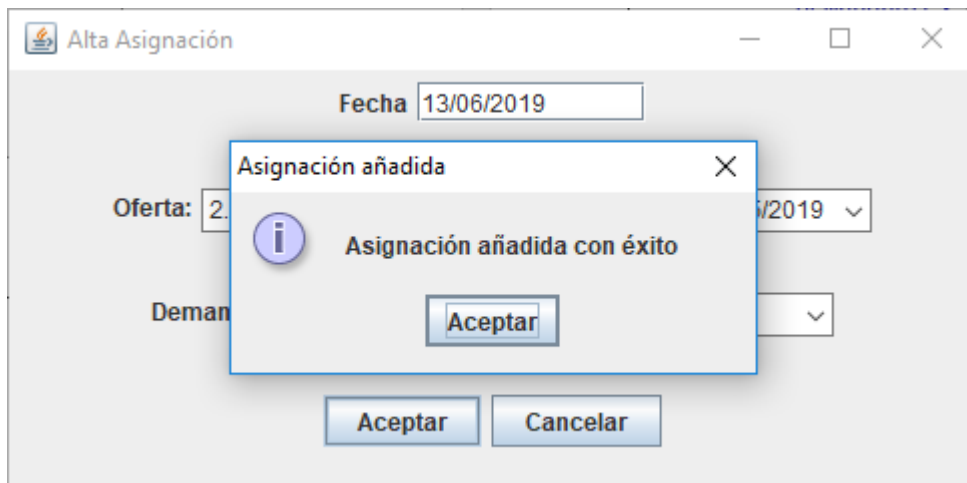
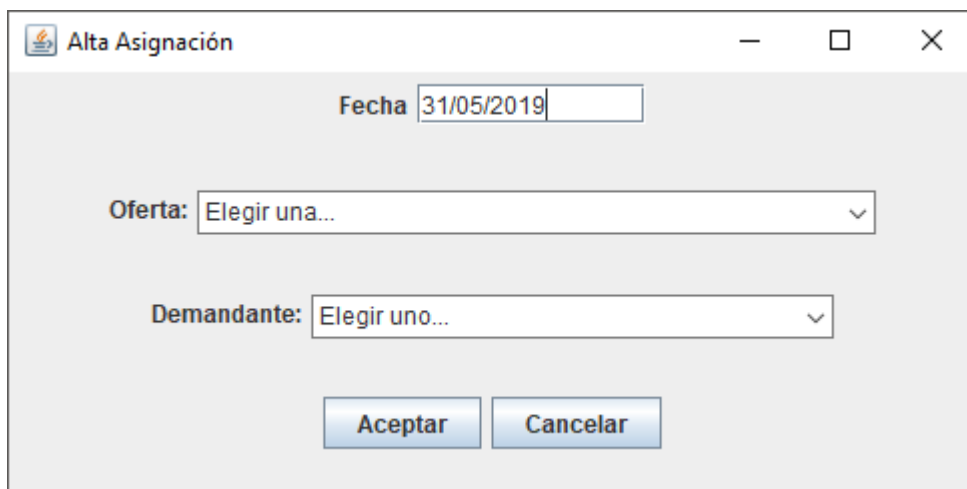
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    DefaultTableModel modelo = new DefaultTableModel();
    JTable tablaOfertas= new JTable(modelo);
    JButton btnAceptar = new JButton("Aceptar");
    JPanel pnlB = new JPanel();
    public ConsultaOfertas() {
        this.setSize(500,200);
        this.setTitle("Consulta Ofertas");
        this.setLocationRelativeTo(null);
        this.setLayout(new BorderLayout());
        this.add(new
JScrollPane(tablaOfertas),BorderLayout.CENTER);

```

```
        pnlB.add(btnAceptar);  
        this.add(pnlB, BorderLayout.SOUTH);  
        this.setVisible(true);  
    }  
}
```

8. Gestión: Alta

Para el Alta de la gestión (tabla que relaciona los demandantes y las ofertas) encontramos una clase llamada AltaAsignacion en la que asignamos una oferta con un demandante a través de dos Choice.



Código AltaAsignacion:

```
package es.studium.PracticaMVC;

import java.awt.Choice;
import java.awt.GridLayout;
import javax.swing.*;

public class AltaAsignacion extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    Choice ofertas = new Choice();
    Choice demandantes = new Choice();

    JLabel lblFecha = new JLabel("Fecha");
    JLabel lblOferta = new JLabel("Oferta:");
    JLabel lblDemandante = new JLabel("Demandante:");

    JTextField txtFecha = new JTextField(10);

    JButton btnAceptar = new JButton("Aceptar");
    JButton btnCancelar = new JButton("Cancelar");

    JPanel pnl1 = new JPanel();
    JPanel pnl2 = new JPanel();
    JPanel pnl3 = new JPanel();
    JPanel pnl4 = new JPanel();

    public AltaAsignacion() {
        this.setSize(500,250);
        this.setTitle("Alta Asignación");
        this.setLocationRelativeTo(null);
        this.setLayout(new GridLayout(4,1));

        ofertas.add("Elegir una...");
        demandantes.add("Elegir uno...");

        pnl1.add(lblFecha);
        pnl1.add(txtFecha);
        pnl2.add(lblOferta);
        pnl2.add(ofertas);
        pnl3.add(lblDemandante);
        pnl3.add(demandantes);
    }
}
```

```
        pnl4.add(btnAceptar);  
        pnl4.add(btnCancelar);  
  
        this.add(pnl1);  
        this.add(pnl2);  
        this.add(pnl3);  
        this.add(pnl4);  
  
        this.setVisible(true);  
    }  
}
```


9. Bibliografía

Entornos de Desarrollo. (10 de 06 de 2019). Temario de Entornos de desarrollo tema 5 Modelo Vista Controlador.
http://aulastudium.com/pluginfile.php?file=%2F20650%2Fmod_resource%2Fcontent%2F0%2FED%20T5%20Temario.pdf