Patrón de diseño de un proyecto

Integrantes:

- Changanaqui Velasquez, Leonardo Daniel

Docente:

ARIANA MAYBEE ORUE MEDINA

Patrón MVC

El patrón MVC, que significa Modelo-Vista-Controlador, es un diseño arquitectónico utilizado comúnmente en el desarrollo de software para separar las preocupaciones y organizar el código de manera más eficiente. Este patrón divide una aplicación en tres componentes principales:

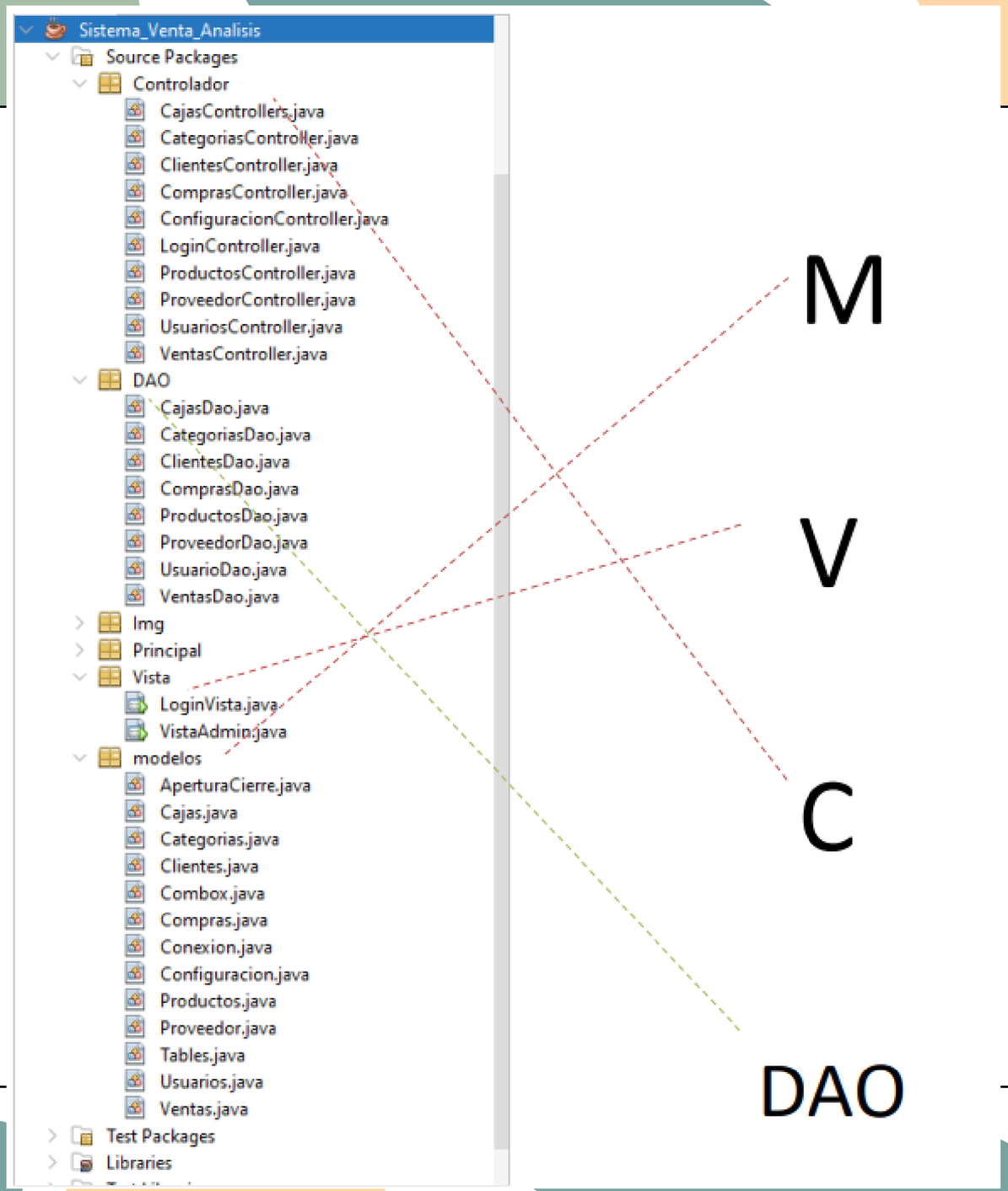
- Modelo (Model): Representa la estructura de datos y lógica de negocio de la aplicación. Es responsable de gestionar los datos, realizar operaciones sobre ellos y notificar a las vistas sobre cualquier cambio en los datos.
- Vista (View): Es responsable de la presentación de la información al usuario y de recibir las interacciones del usuario. La vista muestra los datos al usuario y, en algunos casos, también permite la modificación de los datos.
- Controlador (Controller): Actúa como intermediario entre el modelo y la vista. Responde a los eventos generados por la interfaz de usuario y actualiza el modelo en consecuencia. También actualiza la vista cuando el modelo cambia.

Patrón DAO

El patrón DAO (Data Access Object) es un patrón de diseño que se utiliza en el desarrollo de software para abstraer y encapsular el acceso a la fuente de datos, como una base de datos. Su propósito principal es separar la lógica de acceso a los datos de la lógica de negocio.

- Data Access Object (DAO): Es una interfaz o clase abstracta que define los métodos para acceder a la fuente de datos. Esta interfaz proporciona una capa de abstracción entre la lógica de negocio y el acceso a los datos. Los métodos en la interfaz DAO suelen incluir operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y otras operaciones relacionadas con el acceso a datos.
- Implementación del DAO: Clases que implementan la interfaz DAO y proporcionan la lógica concreta para interactuar con la fuente de datos. Implica la escritura y ejecución de consultas SQL en el caso de bases de datos relacionales, llamadas a API en el caso de servicios web, o cualquier otro mecanismo de acceso a datos específico.
- Objetos de Transferencia de Datos (DTO): En algunos casos, el patrón DAO utiliza objetos de transferencia de datos para representar la información que se va a leer o escribir en la fuente de datos. Estos objetos suelen ser simples contenedores de datos sin lógica de negocio.

Paquetes del proyecto



M

V

C

DAO

Capa del modelo

```java
package modelos;

public class Clientes {
    private int id;
    private String nombre;
    private String telefono;
    private String direccion;
    private String estado;

    public Clientes(){

    }
    public Clientes(int id, String nombre, String telefono, String direccion, String estado) {
        this.id = id;
        this.nombre = nombre;
        this.telefono = telefono;
        this.direccion = direccion;
        this.estado = estado;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

}
```

Capa controlador

```java
package Controlador;

import Vista.VistaAdmin;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import modelos.Clientes;
import DAO.ClientesDao;
import modelos.Combox;
import modelos.Tables;
import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;

public class ClientesController implements ActionListener ,MouseListener,KeyListener{
    private Clientes cl;
    private ClientesDao clDao;
    private VistaAdmin vista;
    DefaultTableModel modelo = new DefaultTableModel();
```

```java
public ClientesController(Clientes cl, ClientesDao clDao, VistaAdmin vista) {
    this.cl = cl;
    this.clDao = clDao;
    this.vista = vista;
    this.vista.btnRegistrarCli.addActionListener(l: this);
    this.vista.btnModificarCli.addActionListener(l: this);
    this.vista.btnNuevoCli.addActionListener(l: this);
    this.vista.TableClientes.addMouseListener(l: this);
    this.vista.jMenuEliminarCli.addActionListener(l: this);
    this.vista.jMenuReingresarCli.addActionListener(l: this);
    this.vista.txtBuscarCli.addKeyListener(l: this);
    this.vista.jLabelClientes.addMouseListener(l: this);
    listarClientes();
    llenarClientes();
    AutoCompleteDecorator.decorate(vista.cbxClientesVentas);
}

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == vista.btnRegistrarCli){
        if(vista.txtNombreCli.getText().equals(anObject: "") ||
                vista.txtTelefonoCli.getText().equals(anObject: "")
                || vista.txtDireccionCli.getText().equals(anObject: "")){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Todos los campos son obligatorios");
        }else{
            cl.setNombre(nombre:vista.txtNombreCli.getText());
            cl.setTelefono(telefono: vista.txtTelefonoCli.getText());
            cl.setDireccion(direccion:vista.txtDireccionCli.getText());
            String respuesta = clDao.registrar(cl);
            switch (respuesta) {
                case "existe":
                    JOptionPane.showMessageDialog(parentComponent: null,message: "El telefono del Cliente Repite");
                    break;
                case "Registrado":
                    limpiarTable();
```

```java
public void listarClientes(){
    Tables color = new Tables();
    vista.TableClientes.setDefaultRenderer(columnClass: vista.TableClientes.getColumnClass(column:0), renderer: color);
    List<Clientes> lista = clDao.ListaCliente(valor: vista.txtBuscarCli.getText());
    modelo = (DefaultTableModel) vista.TableClientes.getModel();
    Object[] ob = new Object[5];
    for (int i=0; i<lista.size();i++){
        ob[0] = lista.get(index: i).getId();
        ob[1] = lista.get(index: i).getNombre();
        ob[2] = lista.get(index: i).getTelefono();
        ob[3] = lista.get(index: i).getDireccion();
        ob[4] = lista.get(index: i).getEstado();
        modelo.addRow(rowData: ob);
    }
    vista.TableClientes.setModel(dataModel:modelo);
}

public void limpiarTable(){
    for(int i = 0;i<modelo.getRowCount();i++){
        modelo.removeRow(row: i);
        i = i-1;
    }
}
```

Capa DAO

```java
package DAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import modelos.Clientes;
import modelos.Conexion;

public class ClientesDao {
    Conexion cn =new Conexion();
    Connection con;
    PreparedStatement ps;
    ResultSet rs;

     public String registrar(Clientes cl){
        String consulta = "SELECT * FROM clientes WHERE telefono = ?" ;
        String sql= "INSERT INTO clientes (nombre, telefono,direccion) VALUES (?,?,?)";
        try{
            con= cn.getConexion();
            ps = con.prepareStatement(sql: consulta);
            ps.setString(parameterIndex: 1, x: cl.getTelefono());
            rs = ps.executeQuery();
            if(rs.next()){
                return "existe";
            }else{
            ps = con.prepareStatement(sql);
            ps.setString(parameterIndex: 1, x: cl.getNombre());
            ps.setString(parameterIndex: 2, x: cl.getTelefono());
            ps.setString(parameterIndex: 3, x: cl.getDireccion());
            ps.execute();
            return "Registrado";
```

```
            return "Registrado";
            }
        }catch(SQLException e){
            JOptionPane.showMessageDialog(parentComponent: null, message: e.toString());
            return "Error";
        }
    }

    public List ListaCliente(String valor){
        List<Clientes> listaCli = new ArrayList();
        String sql = "SELECT * FROM clientes ORDER BY estado ASC";
        String buscar = "SELECT * FROM clientes WHERE nombre LIKE '%"+valor+"%' OR telefono LIKE '%"+valor+"%'";
        try{
            con = cn.getConexion();
            if (valor.equalsIgnoreCase(anotherString: "")){
             ps = con.prepareStatement(sql);
             rs = ps.executeQuery();
            }else{
             ps = con.prepareStatement(sql: buscar);
             rs = ps.executeQuery();
            }
            while(rs.next()){
                Clientes cl = new Clientes();
                cl.setId(id: rs.getInt(columnLabel: "id"));
                cl.setNombre(nombre: rs.getString(columnLabel: "nombre"));
                cl.setTelefono(telefono: rs.getString(columnLabel: "telefono"));
                cl.setDireccion(direccion: rs.getString(columnLabel: "direccion"));
                cl.setEstado(estado: rs.getString(columnLabel: "estado"));
                listaCli.add(e: cl);
            }
        }catch (SQLException e){
            JOptionPane.showMessageDialog(parentComponent: null,message: e.toString());
        }
        return listaCli;
    }
```

```
    public String modificar(Clientes cl){
        String consulta = "SELECT * FROM clientes WHERE telefono = ? AND id != ?" ;
        String sql= "UPDATE clientes SET nombre =?, telefono=?, direccion=? WHERE id=?";
        try{
            con= cn.getConexion();
            ps = con.prepareStatement(sql: consulta);
            ps.setString(parameterIndex: 1, x: cl.getTelefono());
            ps.setInt(parameterIndex: 2, x: cl.getId());
            rs = ps.executeQuery();
            if(rs.next()){
                return "existe";
            }else{
            ps = con.prepareStatement(sql);
            ps.setString(parameterIndex: 1, x: cl.getNombre());
            ps.setString(parameterIndex: 2, x: cl.getTelefono());
            ps.setString(parameterIndex: 3, x: cl.getDireccion());
            ps.setInt(parameterIndex: 4, x: cl.getId());
            ps.execute();
            return "modificado";
            }
        }catch(SQLException e){
            JOptionPane.showMessageDialog(parentComponent: null, message: e.toString());
            return "Error";
        }
    }
```

```
    public boolean accion(String estado, int id){
        String sql= "UPDATE clientes SET estado = ? WHERE id=?";
        try {
            con = cn.getConexion();
            ps = con.prepareStatement(sql);
            ps.setString(parameterIndex: 1, x: estado);
            ps.setInt(parameterIndex: 2, x: id);
            ps.execute();
            return true;
        }catch(SQLException e){
            JOptionPane.showMessageDialog(parentComponent: null, message: e.toString());
            return false;
        }
    }
}
```

Gracias