JAVA - Ejercicio 9

CAMPUSFP

DIEGO CUADRADO

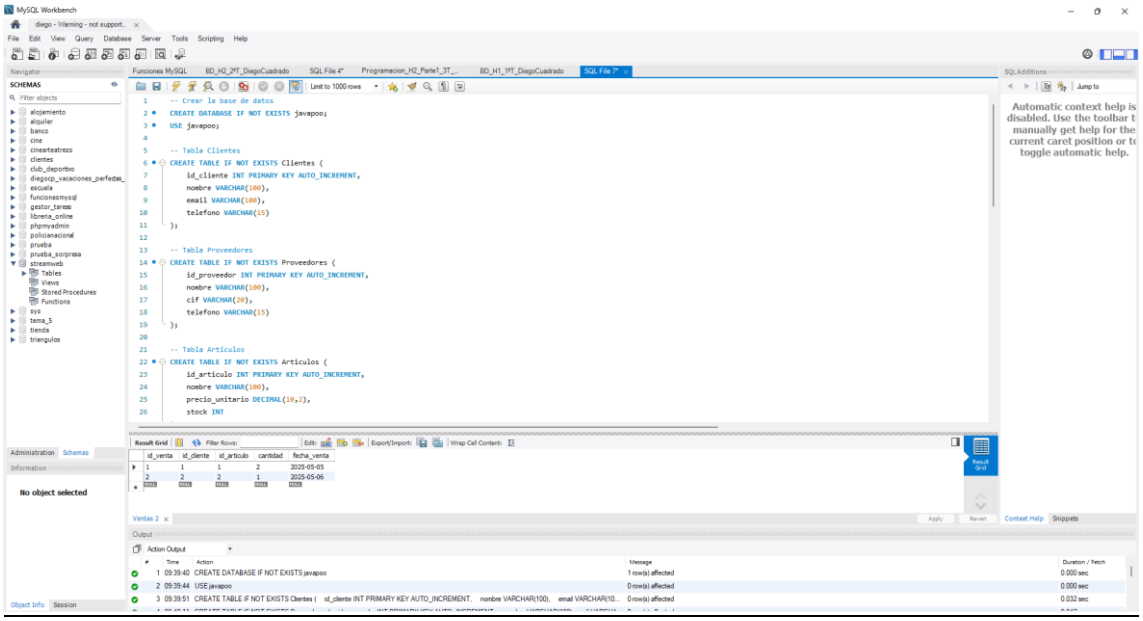
Ejercicio Práctico: Gestión de la Empresa JAVAPOO S.L.

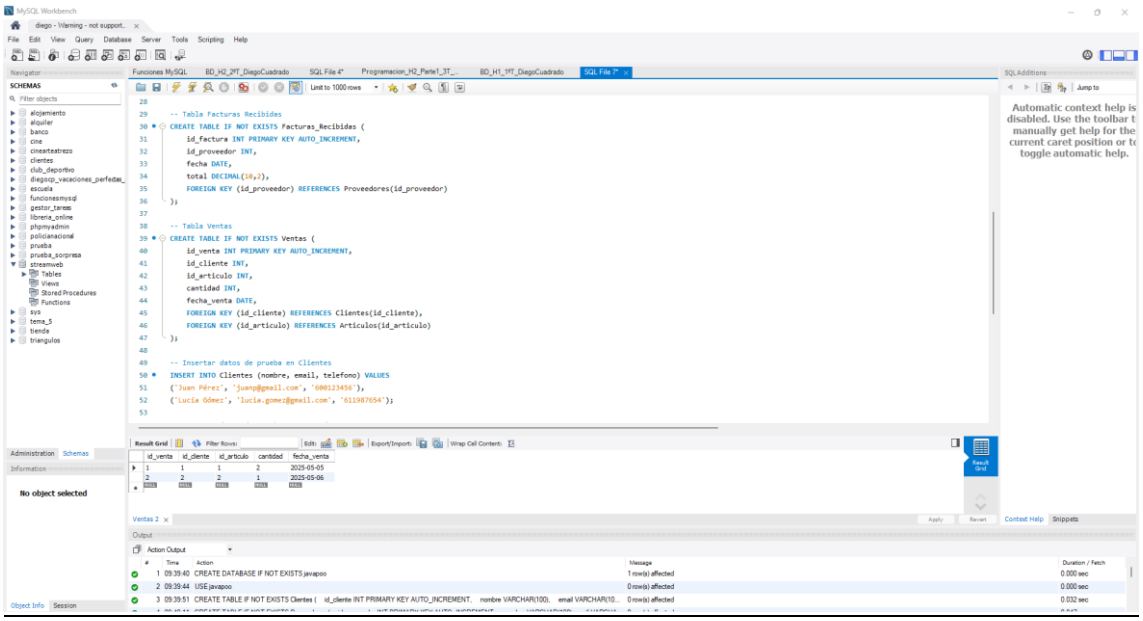
Contexto del ejercicio: La empresa ficticia **JAVAPOO S.L.** se dedica a la compraventa de artículos informáticos. Tu misión será desarrollar una **aplicación de consola en Java** que permita la gestión completa de la empresa a través de **menús de texto**, implementando operaciones CRUD (Crear, Leer, Actualizar, Borrar) para todas las entidades de la base de datos.

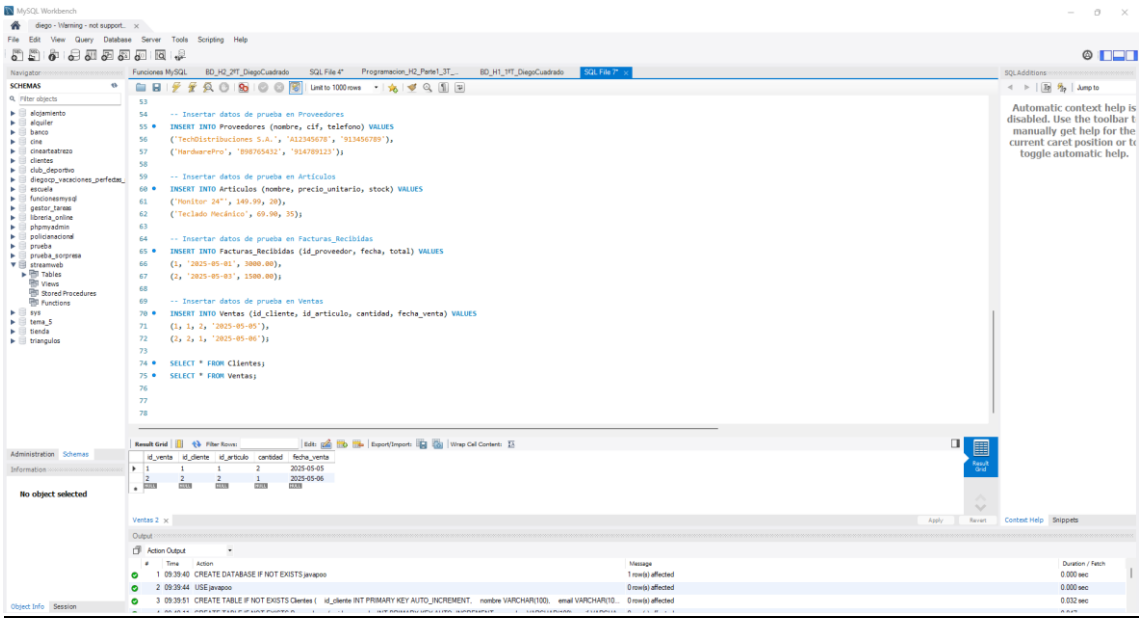
ENLACE A GITHUB:

https://github.com/DiegoCuadrado/PROGRAMACION.git

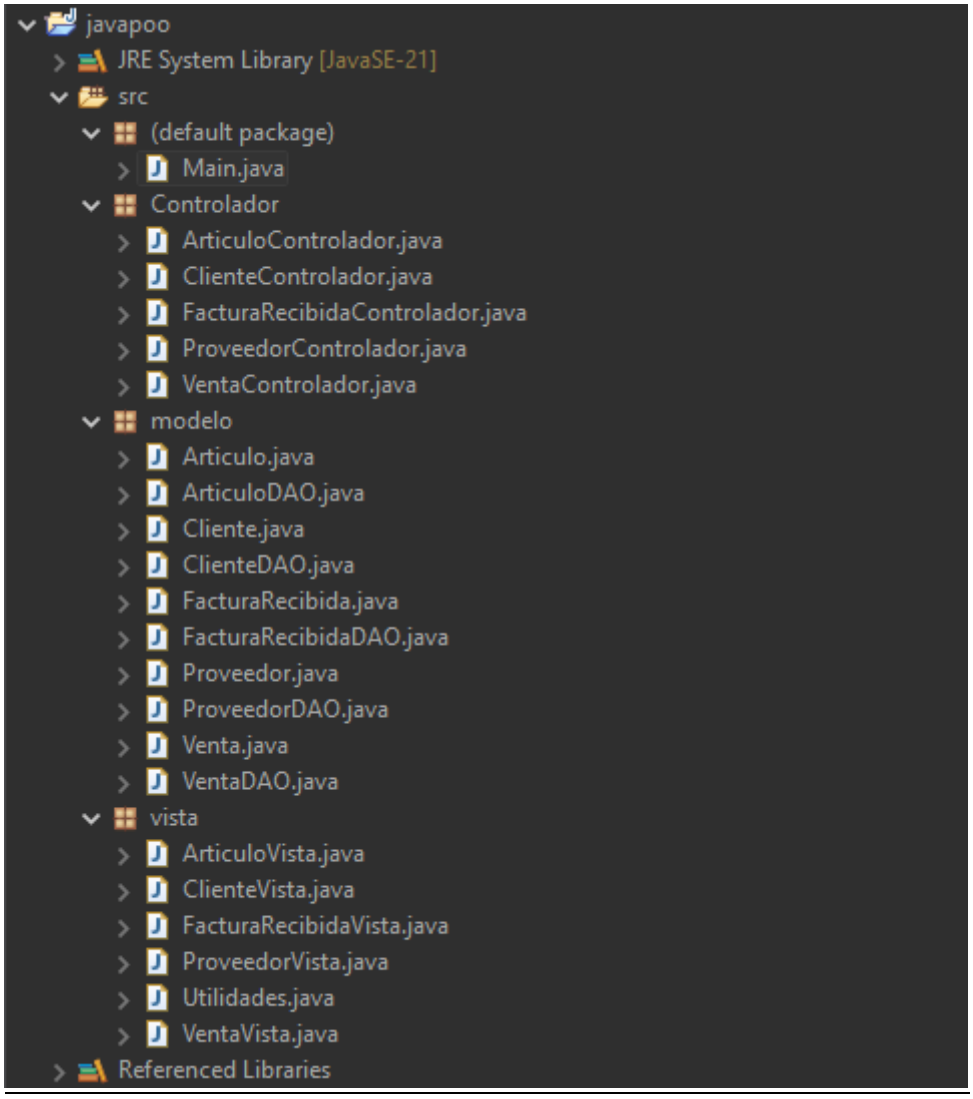
PARTE 1: Creación de la base de datos







PARTE 3: Desarrollo de la aplicación Java



Main.java:

```
import Controlador.*;

import modelo.*;

import vista.*;


import java.util.List;


public class Main {

    public static void main(String[] args) {

        // Crear las instancias de las vistas y los DAOs

        ClienteVista clienteVista = new ClienteVista();

        ProveedorVista proveedorVista = new ProveedorVista();

        ArticuloVista articuloVista = new ArticuloVista();

        FacturaRecibidaVista facturaRecibidaVista = new FacturaRecibidaVista();

        VentaVista ventaVista = new VentaVista();


        ClienteDAO clienteDAO = new ClienteDAO();

        ProveedorDAO proveedorDAO = new ProveedorDAO();

        ArticuloDAO articuloDAO = new ArticuloDAO();  // Línea corregida: asegúrate de que exista la clase

        FacturaRecibidaDAO facturaRecibidaDAO = new FacturaRecibidaDAO();

        VentaDAO ventaDAO = new VentaDAO();  // Se pasa ArticuloDAO al constructor


        // Crear los controladores para cada entidad

        ClienteControlador clienteControlador = new ClienteControlador(clienteVista, clienteDAO);

        ProveedorControlador proveedorControlador = new ProveedorControlador(proveedorVista, proveedorDAO);
```

```
        ArticuloControlador articuloControlador = new ArticuloControlador(articuloVista, articuloDAO);

        FacturaRecibidaControlador facturaRecibidaControlador = new FacturaRecibidaControlador(facturaRecibidaVista, facturaRecibidaDAO);

        VentaControlador ventaControlador = new VentaControlador(ventaVista, ventaDAO);



        // Menú principal de la aplicación

        int opcion;

        do {

            System.out.println("\n=== MENÚ PRINCIPAL ===");

            System.out.println("1. Gestión de Clientes");

            System.out.println("2. Gestión de Proveedores");

            System.out.println("3. Gestión de Artículos");

            System.out.println("4. Gestión de Facturas Recibidas");

            System.out.println("5. Gestión de Ventas");

            System.out.println("6. Informes de Ventas por Cliente");

            System.out.println("0. Salir");

            System.out.print("Seleccione una opción: ");

            opcion = Utilidades.leerInt();



            switch (opcion) {

                case 1:

                    clienteControlador.gestionarClientes();  // Línea corregida: asegúrate de que el método exista

                    break;

                case 2:

                    proveedorControlador.gestionarProveedores();

                    break;
```

```
            case 3:
                articuloControlador.gestionarArticulos();
                break;
            case 4:
                facturaRecibidaControlador.gestionarFacturasRecibidas();
                break;
            case 5:
                ventaControlador.gestionarVentas();
                break;
            case 6:
                generarInformeVentasPorCliente(ventaDAO);
                break;
            case 0:
                System.out.println("Gracias por usar la aplicación. ¡Hasta luego!");
                break;
            default:
                System.out.println("Opción no válida. Por favor, intente de nuevo.");
        }
    } while (opcion != 0);
}

private static void generarInformeVentasPorCliente(VentaDAO ventaDAO) {
    System.out.println("\n=== Informe de Ventas por Cliente ===");
    System.out.print("Ingrese el ID del cliente para generar el informe: ");
    int idCliente = Utilidades.leerInt();

    List<Venta> ventas = ventaDAO.obtenerVentasPorCliente(idCliente); // Este
método debe existir
```

```
        if (ventas.isEmpty()) {
            System.out.println("El cliente con ID " + idCliente + " no tiene ventas registradas.");
        } else {
            double totalGastado = 0;
            System.out.println("Ventas de Cliente ID " + idCliente + ":");
            for (Venta venta : ventas) {
                Articulo articulo = venta.getArticulo(new ArticuloDAO());  // Obtener artículo utilizando ArticuloDAO
                System.out.println("Artículo: " + articulo.getNombre() +
                        ", Cantidad: " + venta.getCantidad() +
                        ", Fecha: " + venta.getFechaVenta());
                totalGastado += venta.getCantidad() * articulo.getPrecioUnitario();
            }
            System.out.println("Total gastado: €" + totalGastado);
        }
    }
}
```

PAQUETE CONTROLADOR:

ARTICULOCONTROLADOR.JAVA:

```
package Controlador;

import modelo.Articulo;
import modelo.ArticuloDAO;
import vista.ArticuloVista;

import java.util.List;
```

```
public class ArticuloControlador {
    private ArticuloVista vista;
    private ArticuloDAO dao;

    public ArticuloControlador(ArticuloVista vista, ArticuloDAO dao) {
        this.vista = vista;
        this.dao = dao;
    }

    public void gestionarArticulos() {
        int opcion;
        do {
            opcion = vista.mostrarMenu();
            switch (opcion) {
                case 1:
                    listarArticulos();
                    break;
                case 2:
                    agregarArticulo();
                    break;
                case 3:
                    modificarArticulo();
                    break;
                case 4:
                    eliminarArticulo();
                    break;
```

```
            case 0:
                System.out.println("Volviendo al menú principal...");
                break;
            default:
                vista.mostrarError("Opción no válida.");
        }
    } while (opcion != 0);
}

private void listarArticulos() {
    List<Articulo> articulos = dao.obtenerTodos();
    if (articulos.isEmpty()) {
        vista.mostrarMensaje("No hay artículos registrados.");
    } else {
        vista.mostrarArticulos(articulos);
    }
}

private void agregarArticulo() {
    Articulo nuevoArticulo = vista.solicitarDatosArticulo();
    boolean exito = dao.agregar(nuevoArticulo);
    if (exito) {
        vista.mostrarMensaje("Artículo agregado correctamente.");
    } else {
        vista.mostrarError("No se pudo agregar el artículo.");
    }
}
```

```
    private void modificarArticulo() {
        int idArticulo = vista.solicitarIdArticulo();
        Articulo articuloExistente = dao.obtenerPorId(idArticulo);
        if (articuloExistente != null) {
            Articulo articuloModificado =
vista.solicitarDatosModificacion(articuloExistente);
            boolean exito = dao.modificar(articuloModificado);
            if (exito) {
                vista.mostrarMensaje("Artículo modificado correctamente.");
            } else {
                vista.mostrarError("No se pudo modificar el artículo.");
            }
        } else {
            vista.mostrarError("El artículo con ID " + idArticulo + " no existe.");
        }
    }

    private void eliminarArticulo() {
        int idArticulo = vista.solicitarIdArticulo();
        boolean exito = dao.eliminar(idArticulo);
        if (exito) {
            vista.mostrarMensaje("Artículo eliminado correctamente.");
        } else {
            vista.mostrarError("No se pudo eliminar el artículo.");
        }
    }
}
```

CLIENTECONTROLADOR.JAVA:

```
package Controlador;

import modelo.Cliente;
import modelo.ClienteDAO;
import vista.ClienteVista;
import vista.Utilidades;

import java.util.List;

public class ClienteControlador {

    private ClienteDAO clienteDAO;
    private ClienteVista clienteVista;

    // Constructor con parámetros (para que funcione con Main.java)
    public ClienteControlador(ClienteVista clienteVista, ClienteDAO clienteDAO) {
        this.clienteDAO = clienteDAO;
        this.clienteVista = clienteVista;
    }

    // Método para gestionar menú de clientes
    public void gestionarClientes() {
        int opcion;
        do {
            System.out.println("\n--- Gestión de Clientes ---");
            System.out.println("1. Listar clientes");
```

```
        System.out.println("2. Agregar cliente");
        System.out.println("3. Modificar cliente");
        System.out.println("4. Eliminar cliente");
        System.out.println("0. Volver al menú principal");
        System.out.print("Seleccione una opción: ");
        opcion = Utilidades.leerInt();

        switch (opcion) {
            case 1:
                listarClientes();
                break;
            case 2:
                agregarCliente();
                break;
            case 3:
                modificarCliente();
                break;
            case 4:
                eliminarCliente();
                break;
            case 0:
                System.out.println("Volviendo al menú principal...");
                break;
            default:
                System.out.println("Opción inválida.");
        }
    } while (opcion != 0);
```

```
}

public void agregarCliente() {
    String nombre = clienteVista.leerNombre();
    String email = clienteVista.leerEmail();
    String telefono = clienteVista.leerTelefono();

    Cliente cliente = new Cliente(nombre, email, telefono);

    if (clienteDAO.agregar(cliente)) {
        clienteVista.mostrarMensaje("Cliente agregado correctamente.");
    } else {
        clienteVista.mostrarMensaje("Error al agregar el cliente.");
    }
}

public void listarClientes() {
    List<Cliente> clientes = clienteDAO.obtenerTodos();
    clienteVista.mostrarClientes(clientes);
}

public void modificarCliente() {
    int idCliente = clienteVista.leerIdCliente();
    String nombre = clienteVista.leerNombre();
    String email = clienteVista.leerEmail();
    String telefono = clienteVista.leerTelefono();
```

```
        Cliente cliente = new Cliente(idCliente, nombre, email, telefono);

        if (clienteDAO.modificar(cliente)) {
            clienteVista.mostrarMensaje("Cliente modificado correctamente.");
        } else {
            clienteVista.mostrarMensaje("Error al modificar el cliente.");
        }
    }

    public void eliminarCliente() {
        int idCliente = clienteVista.leerIdCliente();

        if (clienteDAO.eliminar(idCliente)) {
            clienteVista.mostrarMensaje("Cliente eliminado correctamente.");
        } else {
            clienteVista.mostrarMensaje("Error al eliminar el cliente.");
        }
    }
}
```

FACTURARECIBIDACONTROLADOR.JAVA:

```
package Controlador;

import modelo.FacturaRecibida;
import modelo.FacturaRecibidaDAO;
import vista.FacturaRecibidaVista;

import java.util.List;
```

```
public class FacturaRecibidaControlador {
    private FacturaRecibidaVista vista;
    private FacturaRecibidaDAO dao;

    // Constructor que recibe la vista y el DAO para la factura recibida
    public FacturaRecibidaControlador(FacturaRecibidaVista vista,
FacturaRecibidaDAO dao) {
        this.vista = vista;
        this.dao = dao;
    }

    // Método principal para gestionar las facturas recibidas
    public void gestionarFacturasRecibidas() {
        int opcion;
        do {
            opcion = vista.mostrarMenu();  // Muestra el menú de opciones
            switch (opcion) {
                case 1:
                    listarFacturas();  // Lista todas las facturas
                    break;
                case 2:
                    agregarFactura();  // Agrega una nueva factura
                    break;
                case 3:
                    modificarFactura();  // Modifica una factura existente
                    break;
                case 4:
```

```
                eliminarFactura();  // Elimina una factura
                break;
            case 0:
                System.out.println("Volviendo al menú principal...");  // Opción de salir
                break;
            default:
                vista.mostrarError("Opción no válida.");  // Si la opción es inválida
        }
    } while (opcion != 0);  // Repite hasta que se elija la opción 0
}

// Método para listar las facturas
private void listarFacturas() {
    List<FacturaRecibida> facturas = dao.obtenerTodas();  // Obtiene todas las facturas del DAO
    if (facturas.isEmpty()) {
        vista.mostrarMensaje("No hay facturas registradas.");
    } else {
        vista.mostrarFacturas(facturas);  // Muestra las facturas en la vista
    }
}

// Método para agregar una nueva factura
private void agregarFactura() {
    FacturaRecibida nuevaFactura = vista.solicitarDatosFactura();  // Solicita los datos de la nueva factura
    boolean exito = dao.agregar(nuevaFactura);  // Intenta agregar la factura al DAO
    if (exito) {
```

```
            vista.mostrarMensaje("Factura agregada correctamente.");
        } else {
            vista.mostrarError("No se pudo agregar la factura.");
        }
    }

    // Método para modificar una factura
    private void modificarFactura() {
        int idFactura = vista.solicitarIdFactura();  // Solicita el ID de la factura a modificar
        FacturaRecibida facturaExistente = dao.obtenerPorId(idFactura);  // Obtiene la factura existente por ID
        if (facturaExistente != null) {
            FacturaRecibida facturaModificada = vista.solicitarDatosModificacion(facturaExistente);  // Solicita los nuevos datos
            boolean exito = dao.modificar(facturaModificada);  // Intenta modificar la factura en el DAO
            if (exito) {
                vista.mostrarMensaje("Factura modificada correctamente.");
            } else {
                vista.mostrarError("No se pudo modificar la factura.");
            }
        } else {
            vista.mostrarError("La factura con ID " + idFactura + " no existe.");
        }
    }

    // Método para eliminar una factura
    private void eliminarFactura() {
        int idFactura = vista.solicitarIdFactura();  // Solicita el ID de la factura a eliminar
```

```
        boolean exito = dao.eliminar(idFactura);  // Intenta eliminar la factura en el DAO
        if (exito) {
            vista.mostrarMensaje("Factura eliminada correctamente.");
        } else {
            vista.mostrarError("No se pudo eliminar la factura.");
        }
    }
}
```

PROVEEDORCONTROLADOR.JAVA:

```
package Controlador;

import modelo.Proveedor;
import modelo.ProveedorDAO;
import vista.ProveedorVista;

import java.util.List;

public class ProveedorControlador {
    private ProveedorVista vista;
    private ProveedorDAO dao;

    public ProveedorControlador(ProveedorVista vista, ProveedorDAO dao) {
        this.vista = vista;
        this.dao = dao;
    }

    public void gestionarProveedores() {
```

```
    int opcion;
    do {
        opcion = vista.mostrarMenu();
        switch (opcion) {
            case 1:
                listarProveedores();
                break;
            case 2:
                agregarProveedor();
                break;
            case 3:
                modificarProveedor();
                break;
            case 4:
                eliminarProveedor();
                break;
            case 0:
                System.out.println("Volviendo al menú principal...");
                break;
            default:
                vista.mostrarError("Opción no válida.");
        }
    } while (opcion != 0);
}

private void listarProveedores() {
    List<Proveedor> proveedores = dao.obtenerTodos();
```

```
        if (proveedores.isEmpty()) {
            vista.mostrarMensaje("No hay proveedores registrados.");
        } else {
            vista.mostrarProveedores(proveedores);
        }
    }

    private void agregarProveedor() {
        Proveedor nuevoProveedor = vista.solicitarDatosProveedor();
        boolean exito = dao.agregar(nuevoProveedor);
        if (exito) {
            vista.mostrarMensaje("Proveedor agregado correctamente.");
        } else {
            vista.mostrarError("No se pudo agregar el proveedor.");
        }
    }

    private void modificarProveedor() {
        int idProveedor = vista.solicitarIdProveedor();
        Proveedor proveedorExistente = dao.obtenerPorId(idProveedor);
        if (proveedorExistente != null) {
            Proveedor proveedorModificado =
vista.solicitarDatosModificacion(proveedorExistente);
            boolean exito = dao.modificar(proveedorModificado);
            if (exito) {
                vista.mostrarMensaje("Proveedor modificado correctamente.");
            } else {
                vista.mostrarError("No se pudo modificar el proveedor.");
```

```
            }
        } else {
            vista.mostrarError("El proveedor con ID " + idProveedor + " no existe.");
        }
    }

    private void eliminarProveedor() {
        int idProveedor = vista.solicitarIdProveedor();
        boolean exito = dao.eliminar(idProveedor);
        if (exito) {
            vista.mostrarMensaje("Proveedor eliminado correctamente.");
        } else {
            vista.mostrarError("No se pudo eliminar el proveedor.");
        }
    }
}
```

VENTACONTROLADOR.JAVA:

```
package Controlador;

import modelo.Venta;
import modelo.VentaDAO;
import vista.VentaVista;

import java.util.List;

public class VentaControlador {
    private VentaVista vista;
```

```
private VentaDAO dao;

public VentaControlador(VentaVista vista, VentaDAO dao) {
    this.vista = vista;
    this.dao = dao;
}

public void gestionarVentas() {
    int opcion;
    do {
        opcion = vista.mostrarMenu();
        switch (opcion) {
            case 1:
                listarVentas();
                break;
            case 2:
                agregarVenta();
                break;
            case 3:
                modificarVenta();
                break;
            case 4:
                eliminarVenta();
                break;
            case 0:
                System.out.println("Volviendo al menú principal...");
                break;
```

```
            default:
                vista.mostrarError("Opción no válida.");
        }
    } while (opcion != 0);
}

private void listarVentas() {
    List<Venta> ventas = dao.obtenerTodas();
    if (ventas.isEmpty()) {
        vista.mostrarMensaje("No hay ventas registradas.");
    } else {
        vista.mostrarVentas(ventas);
    }
}

private void agregarVenta() {
    Venta nuevaVenta = vista.solicitarDatosVenta();
    boolean exito = dao.agregar(nuevaVenta);
    if (exito) {
        vista.mostrarMensaje("Venta agregada correctamente.");
    } else {
        vista.mostrarError("No se pudo agregar la venta.");
    }
}

private void modificarVenta() {
    int idVenta = vista.solicitarIdVenta();
```

```
        Venta ventaExistente = dao.obtenerPorId(idVenta);
        if (ventaExistente != null) {
            Venta ventaModificada = vista.solicitarDatosModificacion(ventaExistente);
            boolean exito = dao.modificar(ventaModificada);
            if (exito) {
                vista.mostrarMensaje("Venta modificada correctamente.");
            } else {
                vista.mostrarError("No se pudo modificar la venta.");
            }
        } else {
            vista.mostrarError("La venta con ID " + idVenta + " no existe.");
        }
    }

    private void eliminarVenta() {
        int idVenta = vista.solicitarIdVenta();
        boolean exito = dao.eliminar(idVenta);
        if (exito) {
            vista.mostrarMensaje("Venta eliminada correctamente.");
        } else {
            vista.mostrarError("No se pudo eliminar la venta.");
        }
    }
}
```

PAQUETE MODELO:

ARTICULO.JAVA:

```java
package modelo;

public class Articulo {
    private int idArticulo;
    private String nombre;
    private double precioUnitario;
    private int stock;

    // Constructor sin ID (para crear artículos nuevos)
    public Articulo(String nombre, double precioUnitario, int stock) {
        this.nombre = nombre;
        this.precioUnitario = precioUnitario;
        this.stock = stock;
    }

    // Constructor con ID (para obtener o modificar artículos existentes)
    public Articulo(int idArticulo, String nombre, double precioUnitario, int
stock) {
        this.idArticulo = idArticulo;
        this.nombre = nombre;
        this.precioUnitario = precioUnitario;
        this.stock = stock;
    }

    // Getters y Setters
    public int getIdArticulo() {
        return idArticulo;
    }

    public void setIdArticulo(int idArticulo) {
        this.idArticulo = idArticulo;
    }

    public String getNombre() {
        return nombre;
    }

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

    public double getPrecioUnitario() {
        return precioUnitario;
    }

    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
```

```
    }

    // Método toString para mostrar la información del artículo
    @Override
    public String toString() {
        return "ID: " + idArticulo + ", Nombre: " + nombre + ", Precio: " +
precioUnitario + ", Stock: " + stock;
    }
}
```

ARTICULODAO.JAVA:

```
package modelo;


import java.sql.*;
import java.util.ArrayList;
import java.util.List;


public class ArticuloDAO {
    private Connection connection;


    public ArticuloDAO(Connection connection) {
        this.connection = connection;
    }
    public ArticuloDAO() {
        try {
            this.connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/javapoo", "root", "1234");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    public boolean agregar(Articulo articulo) {
        String sql = "INSERT INTO articulos (nombre, precio_unitario, stock) VALUES
(?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, articulo.getNombre());
            stmt.setDouble(2, articulo.getPrecioUnitario());
            stmt.setInt(3, articulo.getStock());
            int filasInsertadas = stmt.executeUpdate();
            return filasInsertadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public List<Articulo> obtenerTodos() {
        List<Articulo> articulos = new ArrayList<>();
        String sql = "SELECT * FROM articulos";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                int idArticulo = rs.getInt("id_articulo");
                String nombre = rs.getString("nombre");
                double precioUnitario = rs.getDouble("precio_unitario");
                int stock = rs.getInt("stock");

                articulos.add(new Articulo(idArticulo, nombre, precioUnitario, stock));
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return articulos;
    }

    public Articulo obtenerPorId(int idArticulo) {
        String sql = "SELECT * FROM articulos WHERE id_articulo = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idArticulo);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    String nombre = rs.getString("nombre");
                    double precioUnitario = rs.getDouble("precio_unitario");
                    int stock = rs.getInt("stock");
                    return new Articulo(idArticulo, nombre, precioUnitario, stock);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }

    public boolean modificar(Articulo articulo) {
        String sql = "UPDATE articulos SET nombre = ?, precio_unitario = ?, stock = ?
WHERE id_articulo = ?";
```

```
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, articulo.getNombre());
            stmt.setDouble(2, articulo.getPrecioUnitario());
            stmt.setInt(3, articulo.getStock());
            stmt.setInt(4, articulo.getIdArticulo());
            int filasModificadas = stmt.executeUpdate();
            return filasModificadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean eliminar(int idArticulo) {
        String sql = "DELETE FROM articulos WHERE id_articulo = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idArticulo);
            int filasEliminadas = stmt.executeUpdate();
            return filasEliminadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

CLIENTE.JAVA:

```java
package modelo;

public class Cliente {
    private int idCliente;
    private String nombre;
    private String email;
    private String telefono;

    // Constructor sin idCliente (para agregar nuevos clientes)
    public Cliente(String nombre, String email, String telefono) {
        this.nombre = nombre;
        this.email = email;
        this.telefono = telefono;
    }

    // Constructor con idCliente (para modificar clientes existentes)
    public Cliente(int idCliente, String nombre, String email, String
telefono) {
        this.idCliente = idCliente;
        this.nombre = nombre;
        this.email = email;
        this.telefono = telefono;
    }

    // Getters y Setters
    public int getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(int idCliente) {
        this.idCliente = idCliente;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTelefono() {
        return telefono;
    }

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

```
    // Método toString para mostrar la información del cliente
    @Override
    public String toString() {
        return "ID: " + idCliente + ", Nombre: " + nombre + ", Email: " +
email + ", Teléfono: " + telefono;
    }
}
```

CLIENTEDAO.JAVA:

```
package modelo;


import java.sql.*;
import java.util.ArrayList;
import java.util.List;


public class ClienteDAO {


    private Connection connection;


    // Constructor vacío para inicializar la conexión
    public ClienteDAO() {
        try {
            // Cambia estos datos de conexión según tu configuración
            this.connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/javapoo", "root", "1234");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


    public boolean agregar(Cliente cliente) {
```

```
        String query = "INSERT INTO Clientes (nombre, email, telefono) VALUES (?, ?,
?)";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setString(1, cliente.getNombre());
            statement.setString(2, cliente.getEmail());
            statement.setString(3, cliente.getTelefono());
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public List<Cliente> obtenerTodos() {
        List<Cliente> clientes = new ArrayList<>();
        String query = "SELECT * FROM Clientes";
        try (Statement statement = connection.createStatement();
             ResultSet resultSet = statement.executeQuery(query)) {
            while (resultSet.next()) {
                Cliente cliente = new Cliente(resultSet.getInt("id_cliente"),
resultSet.getString("nombre"),
                        resultSet.getString("email"), resultSet.getString("telefono"));
                clientes.add(cliente);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return clientes;
```

```
    }

    public boolean modificar(Cliente cliente) {
        String query = "UPDATE Clientes SET nombre = ?, email = ?, telefono = ?
WHERE id_cliente = ?";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setString(1, cliente.getNombre());
            statement.setString(2, cliente.getEmail());
            statement.setString(3, cliente.getTelefono());
            statement.setInt(4, cliente.getIdCliente());
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public boolean eliminar(int idCliente) {
        String query = "DELETE FROM Clientes WHERE id_cliente = ?";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setInt(1, idCliente);
            return statement.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }
}
```

FACTURARECIBIDA.JAVA:

```java
package modelo;

public class FacturaRecibida {

    // Atributos
    private int idFactura;
    private String proveedor;
    private String fechaEmision;
    private double montoTotal;

    // Constructor para crear una nueva factura (sin ID)
    public FacturaRecibida(String proveedor, String fechaEmision, double
montoTotal) {
        this.proveedor = proveedor;
        this.fechaEmision = fechaEmision;
        this.montoTotal = montoTotal;
    }

    // Constructor para crear una factura con ID (al obtener de la base de
datos)
    public FacturaRecibida(int idFactura, String proveedor, String
fechaEmision, double montoTotal) {
        this.idFactura = idFactura;
        this.proveedor = proveedor;
        this.fechaEmision = fechaEmision;
        this.montoTotal = montoTotal;
    }

    // Getters y Setters

    public int getIdFactura() {
        return idFactura;
    }

    public void setIdFactura(int idFactura) {
        this.idFactura = idFactura;
    }

    public String getProveedor() {
        return proveedor;
    }

    public void setProveedor(String proveedor) {
        this.proveedor = proveedor;
    }

    public String getFechaEmision() {
        return fechaEmision;
    }

    public void setFechaEmision(String fechaEmision) {
        this.fechaEmision = fechaEmision;
    }

    public double getMontoTotal() {
        return montoTotal;
    }
```

```
    public void setMontoTotal(double montoTotal) {
        this.montoTotal = montoTotal;
    }

    // Método toString para mostrar información de la factura
    @Override
    public String toString() {
        return "Factura ID: " + idFactura + ", Proveedor: " + proveedor + ",
Fecha de Emisión: " + fechaEmision + ", Monto Total: " + montoTotal;
    }
}
```

FACTURARECIBIDADAO.JAVA:

```
package modelo;


import java.sql.*;

import java.util.ArrayList;

import java.util.List;


public class FacturaRecibidaDAO {

   private Connection connection;


   public FacturaRecibidaDAO() {

      // Conectar a la base de datos

      try {

         this.connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/javapoo", "root", "1234");

      } catch (SQLException e) {

         e.printStackTrace();

      }

   }


   // Método para agregar una nueva factura

   public boolean agregar(FacturaRecibida factura) {
```

```
        String sql = "INSERT INTO facturas_recibidas (id_proveedor, fecha_emision,
monto_total) VALUES (?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, factura.getProveedor());
            stmt.setString(2, factura.getFechaEmision());
            stmt.setDouble(3, factura.getMontoTotal());
            int filasInsertadas = stmt.executeUpdate();
            return filasInsertadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Método para obtener todas las facturas
    public List<FacturaRecibida> obtenerTodas() {
        List<FacturaRecibida> facturas = new ArrayList<>();
        String sql = "SELECT * FROM facturas_recibidas";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int id = rs.getInt("id_factura");
                String proveedor = rs.getString("proveedor");
                String fechaEmision = rs.getString("fecha_emision");
                double montoTotal = rs.getDouble("monto_total");
                facturas.add(new FacturaRecibida(id, proveedor, fechaEmision, montoTotal));
            }
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
        return facturas;
    }

    // Método para obtener una factura por ID
    public FacturaRecibida obtenerPorId(int idFactura) {
        FacturaRecibida factura = null;
        String sql = "SELECT * FROM facturas_recibidas WHERE id_factura = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idFactura);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                String proveedor = rs.getString("proveedor");
                String fechaEmision = rs.getString("fecha_emision");
                double montoTotal = rs.getDouble("monto_total");
                factura = new FacturaRecibida(idFactura, proveedor, fechaEmision,
montoTotal);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return factura;
    }

    // Método para modificar una factura
    public boolean modificar(FacturaRecibida factura) {
```

```
        String sql = "UPDATE facturas_recibidas SET proveedor = ?, fecha_emision = ?, monto_total = ? WHERE id_factura = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, factura.getProveedor());
            stmt.setString(2, factura.getFechaEmision());
            stmt.setDouble(3, factura.getMontoTotal());
            stmt.setInt(4, factura.getIdFactura());
            int filasModificadas = stmt.executeUpdate();
            return filasModificadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Método para eliminar una factura por ID
    public boolean eliminar(int idFactura) {
        String sql = "DELETE FROM facturas_recibidas WHERE id_factura = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idFactura);
            int filasEliminadas = stmt.executeUpdate();  // Corregido el error aquí
            return filasEliminadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

PROVEEDOR.JAVA:

```java
package modelo;

public class Proveedor {
    private int idProveedor;
    private String nombre;
    private String direccion;
    private String telefono;

    // Constructor con todos los parámetros
    public Proveedor(int idProveedor, String nombre, String direccion, String
telefono) {
        this.idProveedor = idProveedor;
        this.nombre = nombre;
        this.direccion = direccion;
        this.telefono = telefono;
    }

    // Getters y setters
    public int getIdProveedor() {
        return idProveedor;
    }

    public void setIdProveedor(int idProveedor) {
        this.idProveedor = idProveedor;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getDireccion() {
        return direccion;
    }

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

    public String getTelefono() {
        return telefono;
    }

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

    // ToString para mostrar información de manera más fácil
    @Override
    public String toString() {
        return "ID: " + idProveedor + ", Nombre: " + nombre + ", Dirección: "
+ direccion + ", Teléfono: " + telefono;
    }
}
```

PROVEEDORDAO.JAVA:

```
package modelo;


import java.sql.*;

import java.util.ArrayList;

import java.util.List;


public class ProveedorDAO {

    private Connection connection;


    public ProveedorDAO() {

        try {

            // Asegúrate de colocar la URL, usuario y contraseña correctos de tu base de datos

            this.connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/javapoo", "root", "1234");

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }


    // Método para obtener todos los proveedores

    public List<Proveedor> obtenerTodos() {

        List<Proveedor> proveedores = new ArrayList<>();

        String sql = "SELECT * FROM proveedores"; // Cambia 'proveedores' por el nombre de tu tabla


        try (Statement stmt = connection.createStatement();
```

```
            ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int id = rs.getInt("id_proveedor");
                String nombre = rs.getString("nombre");
                String direccion = rs.getString("direccion");
                String telefono = rs.getString("telefono");
                proveedores.add(new Proveedor(id, nombre, direccion, telefono)); // Aquí cambiamos la instancia de Proveedor
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return proveedores;
    }
    public Proveedor obtenerPorId(int idProveedor) {
        String sql = "SELECT * FROM proveedores WHERE id_proveedor = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idProveedor);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                String nombre = rs.getString("nombre");
                String direccion = rs.getString("direccion");
                String telefono = rs.getString("telefono");
                return new Proveedor(idProveedor, nombre, direccion, telefono);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
        return null;
    }



    // Método para agregar un nuevo proveedor
    public boolean agregar(Proveedor proveedor) {
        String sql = "INSERT INTO proveedores (nombre, direccion, telefono) VALUES
(?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, proveedor.getNombre());
            stmt.setString(2, proveedor.getDireccion());
            stmt.setString(3, proveedor.getTelefono());
            int filasInsertadas = stmt.executeUpdate();
            return filasInsertadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }


    // Método para modificar un proveedor
    public boolean modificar(Proveedor proveedor) {
        String sql = "UPDATE proveedores SET nombre = ?, direccion = ?, telefono = ?
WHERE id_proveedor = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, proveedor.getNombre());
            stmt.setString(2, proveedor.getDireccion());
            stmt.setString(3, proveedor.getTelefono());
```

```
            stmt.setInt(4, proveedor.getIdProveedor());
            int filasModificadas = stmt.executeUpdate();
            return filasModificadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    // Método para eliminar un proveedor
    public boolean eliminar(int idProveedor) {
        String sql = "DELETE FROM proveedores WHERE id_proveedor = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idProveedor);
            int filasEliminadas = stmt.executeUpdate();
            return filasEliminadas > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

VENTA.JAVA:

```java
package modelo;

import java.util.Date;

public class Venta {
    private int idVenta;
    private int idCliente;
    private int idArticulo;
    private int cantidad;
```

```
    private Date fechaVenta;

    public Venta(int idVenta, int idCliente, int idArticulo, int cantidad,
Date fechaVenta) {
        this.idVenta = idVenta;
        this.idCliente = idCliente;
        this.idArticulo = idArticulo;
        this.cantidad = cantidad;
        this.fechaVenta = fechaVenta;
    }

    // Getters
    public int getIdVenta() {
        return idVenta;
    }

    public int getIdCliente() {
        return idCliente;
    }

    public int getIdArticulo() {
        return idArticulo;
    }

    public int getCantidad() {
        return cantidad;
    }

    public Date getFechaVenta() {
        return fechaVenta;
    }

    // Setters
    public void setIdVenta(int idVenta) {
        this.idVenta = idVenta;
    }

    public void setIdCliente(int idCliente) {
        this.idCliente = idCliente;
    }

    public void setIdArticulo(int idArticulo) {
        this.idArticulo = idArticulo;
    }

    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }

    public void setFechaVenta(Date fechaVenta) {
        this.fechaVenta = fechaVenta;
    }

    public Articulo getArticulo(ArticuloDAO articuloDAO) {
        return articuloDAO.obtenerPorId(this.idArticulo);  // Devuelve el
artículo asociado a la venta
    }

    @Override
```

```
    public String toString() {
        return "Venta{" +
                "idVenta=" + idVenta +
                ", idCliente=" + idCliente +
                ", idArticulo=" + idArticulo +
                ", cantidad=" + cantidad +
                ", fechaVenta=" + fechaVenta +
                '}';
    }
}
```

VENTADAO.JAVA:

```
package modelo;


import java.sql.*;
import java.util.ArrayList;
import java.util.List;


public class VentaDAO {
    private Connection connection;


    public VentaDAO() {
        try {
            // Cambia estos valores según tu configuración
            this.connection = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/javapoo",
                    "root",
                    "1234"
            );
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
public List<Venta> obtenerTodas() {
    List<Venta> ventas = new ArrayList<>();
    String sql = "SELECT * FROM ventas";

    try (Statement stmt = connection.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            int idVenta = rs.getInt("id_venta");
            int idCliente = rs.getInt("id_cliente");
            int idArticulo = rs.getInt("id_articulo");
            int cantidad = rs.getInt("cantidad");
            Date fecha = rs.getDate("fecha_venta");

            ventas.add(new Venta(idVenta, idCliente, idArticulo, cantidad, fecha));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return ventas;
}

public List<Venta> obtenerVentasPorCliente(int idCliente) {
```

```
        List<Venta> ventas = new ArrayList<>();
        String sql = "SELECT * FROM ventas WHERE id_cliente=?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idCliente);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                ventas.add(new Venta(
                    rs.getInt("id_venta"),
                    rs.getInt("id_cliente"),
                    rs.getInt("id_articulo"),
                    rs.getInt("cantidad"),
                    rs.getDate("fecha_venta")
                ));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return ventas;
    }



    public boolean agregar(Venta venta) {
        String sql = "INSERT INTO ventas (id_cliente, id_articulo, cantidad, fecha)
VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, venta.getIdCliente());
            stmt.setInt(2, venta.getIdArticulo());
            stmt.setInt(3, venta.getCantidad());
```

```
            stmt.setDate(4, new java.sql.Date(venta.getFechaVenta().getTime()));
            return stmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean modificar(Venta venta) {
        String sql = "UPDATE ventas SET id_cliente = ?, id_articulo = ?, cantidad = ?,
fecha = ? WHERE id_venta = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, venta.getIdCliente());
            stmt.setInt(2, venta.getIdArticulo());
            stmt.setInt(3, venta.getCantidad());
            stmt.setDate(4, new java.sql.Date(venta.getFechaVenta().getTime()));
            stmt.setInt(5, venta.getIdVenta());
            return stmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean eliminar(int idVenta) {
        String sql = "DELETE FROM ventas WHERE id_venta = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idVenta);
```

```
            return stmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public Venta obtenerPorId(int idVenta) {
        String sql = "SELECT * FROM ventas WHERE id_venta = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, idVenta);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                int idCliente = rs.getInt("id_cliente");
                int idArticulo = rs.getInt("id_articulo");
                int cantidad = rs.getInt("cantidad");
                Date fecha = rs.getDate("fecha");

                return new Venta(idVenta, idCliente, idArticulo, cantidad, fecha);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

PAQUETE VISTA:

ARTICULOVISTA.JAVA:

```
package vista;

import modelo.Articulo;

import java.util.List;
import java.util.Scanner;

public class ArticuloVista {
    private Scanner scanner = new Scanner(System.in);

    public int mostrarMenu() {
        System.out.println("\n--- GESTIÓN DE ARTÍCULOS ---");
        System.out.println("1. Listar artículos");
        System.out.println("2. Agregar artículo");
        System.out.println("3. Modificar artículo");
        System.out.println("4. Eliminar artículo");
        System.out.println("0. Volver al menú principal");
        System.out.print("Seleccione una opción: ");
        return scanner.nextInt();
    }

    public Articulo solicitarDatosArticulo() {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nIngrese los datos del artículo:");
        System.out.print("Nombre: ");
```

```
        String nombre = scanner.nextLine();
        System.out.print("Precio unitario: ");
        double precio = scanner.nextDouble();
        System.out.print("Stock: ");
        int stock = scanner.nextInt();

        return new Articulo(0, nombre, precio, stock);
    }

    public int solicitarIdArticulo() {
        System.out.print("Ingrese el ID del artículo: ");
        return scanner.nextInt();
    }

    public Articulo solicitarDatosModificacion(Articulo articulo) {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nModificando artículo: " + articulo.getIdArticulo());
        System.out.print("Nuevo nombre [" + articulo.getNombre() + "]: ");
        String nombre = scanner.nextLine();
        if (!nombre.isBlank()) articulo.setNombre(nombre);

        System.out.print("Nuevo precio unitario [" + articulo.getPrecioUnitario() + "]: ");
        String precioStr = scanner.nextLine();
        if (!precioStr.isBlank())
articulo.setPrecioUnitario(Double.parseDouble(precioStr));

        System.out.print("Nuevo stock [" + articulo.getStock() + "]: ");
        String stockStr = scanner.nextLine();
```

```
        if (!stockStr.isBlank()) articulo.setStock(Integer.parseInt(stockStr));

        return articulo;
    }

    public void mostrarArticulos(List<Articulo> articulos) {
        System.out.println("\n--- LISTADO DE ARTÍCULOS ---");
        for (Articulo a : articulos) {
            System.out.println(a);
        }
    }

    public void mostrarMensaje(String mensaje) {
        System.out.println(mensaje);
    }

    public void mostrarError(String mensaje) {
        System.err.println("ERROR: " + mensaje);
    }
}
```

CLIENTEVISTA.JAVA:

```
package vista;

import modelo.Cliente;

import java.util.List;
import java.util.Scanner;
```

```
public class ClienteVista {

    private Scanner scanner;

    public ClienteVista() {
        this.scanner = new Scanner(System.in);
    }

    // Mostrar el menú de opciones
    public int mostrarMenu() {
        System.out.println("---- Menú de Gestión de Clientes ----");
        System.out.println("1. Listar Clientes");
        System.out.println("2. Agregar Cliente");
        System.out.println("3. Modificar Cliente");
        System.out.println("4. Eliminar Cliente");
        System.out.println("0. Volver al menú principal");
        System.out.print("Selecciona una opción: ");
        return scanner.nextInt();
    }

    // Métodos para leer los datos del cliente desde la consola
    public String leerNombre() {
        System.out.print("Introduce el nombre del cliente: ");
        return scanner.next();
    }
```

```
public String leerEmail() {
    System.out.print("Introduce el email del cliente: ");
    return scanner.next();
}

public String leerTelefono() {
    System.out.print("Introduce el teléfono del cliente: ");
    return scanner.next();
}

public int leerIdCliente() {
    System.out.print("Introduce el ID del cliente: ");
    return scanner.nextInt();
}

// Métodos para mostrar los resultados
public void mostrarClientes(List<Cliente> clientes) {
    if (clientes.isEmpty()) {
        System.out.println("No hay clientes registrados.");
    } else {
        System.out.println("---- Lista de Clientes ----");
        for (Cliente cliente : clientes) {
            System.out.println(cliente);
        }
    }
}
```

```
    // Mostrar mensajes
    public void mostrarMensaje(String mensaje) {
        System.out.println(mensaje);
    }
}
```

FACTURARECIBIDAVISTA.JAVA:

```
package vista;

import modelo.FacturaRecibida;

import java.util.List;
import java.util.Scanner;

public class FacturaRecibidaVista {
    private Scanner scanner = new Scanner(System.in);

    // Método para mostrar el menú de opciones
    public int mostrarMenu() {
        System.out.println("\n--- GESTIÓN DE FACTURAS RECIBIDAS ---");
        System.out.println("1. Listar facturas");
        System.out.println("2. Agregar factura");
        System.out.println("3. Modificar factura");
        System.out.println("4. Eliminar factura");
        System.out.println("0. Volver al menú principal");
        System.out.print("Seleccione una opción: ");
        return scanner.nextInt();
    }
```

```
    // Método para solicitar los datos para agregar una factura
    public FacturaRecibida solicitarDatosFactura() {
        scanner.nextLine();  // Limpiar buffer
        System.out.println("\nIngrese los datos de la factura:");
        System.out.print("Proveedor: ");
        String proveedor = scanner.nextLine();
        System.out.print("Fecha de emisión (YYYY-MM-DD): ");
        String fechaEmision = scanner.nextLine();
        System.out.print("Monto total: ");
        double montoTotal = scanner.nextDouble();

        // Crear y devolver un objeto FacturaRecibida
        return new FacturaRecibida(proveedor, fechaEmision, montoTotal);
    }

    // Método para solicitar el ID de la factura (para modificaciones o eliminaciones)
    public int solicitarIdFactura() {
        System.out.print("Ingrese el ID de la factura: ");
        return scanner.nextInt();
    }

    // Método para mostrar las facturas listadas
    public void mostrarFacturas(List<FacturaRecibida> facturas) {
        System.out.println("\n--- LISTADO DE FACTURAS RECIBIDAS ---");
        for (FacturaRecibida factura : facturas) {
            System.out.println(factura);  // Asegúrate de que FacturaRecibida tenga un buen método toString()
```

```
    }
}

// Método para mostrar mensajes informativos
public void mostrarMensaje(String mensaje) {
    System.out.println(mensaje);
}

// Método para mostrar mensajes de error
public void mostrarError(String mensaje) {
    System.err.println("ERROR: " + mensaje);
}

// Método para solicitar modificaciones en una factura existente
public FacturaRecibida solicitarDatosModificacion(FacturaRecibida factura) {
    scanner.nextLine();  // Limpiar buffer
    System.out.println("\nModificando factura ID: " + factura.getIdFactura());

    // Solicitar nuevos datos
    System.out.print("Nuevo proveedor [" + factura.getProveedor() + "]: ");
    String proveedor = scanner.nextLine();
    if (!proveedor.isBlank()) factura.setProveedor(proveedor);

    System.out.print("Nueva fecha de emisión [" + factura.getFechaEmision() + "]: ");
    String fechaEmision = scanner.nextLine();
    if (!fechaEmision.isBlank()) factura.setFechaEmision(fechaEmision);
```

```
        System.out.print("Nuevo monto total [" + factura.getMontoTotal() + "]: ");
        double montoTotal = scanner.nextDouble();
        if (montoTotal > 0) factura.setMontoTotal(montoTotal);

        return factura;
    }
}
```

PROVEEDORVISTA.JAVA:

```
package vista;

import modelo.Proveedor;

import java.util.List;
import java.util.Scanner;

public class ProveedorVista {
    private Scanner scanner = new Scanner(System.in);

    public int mostrarMenu() {
        System.out.println("\n--- GESTIÓN DE PROVEEDORES ---");
        System.out.println("1. Listar proveedores");
        System.out.println("2. Agregar proveedor");
        System.out.println("3. Modificar proveedor");
        System.out.println("4. Eliminar proveedor");
        System.out.println("0. Volver al menú principal");
        System.out.print("Seleccione una opción: ");
        return scanner.nextInt();
```

```
    }

    public Proveedor solicitarDatosProveedor() {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nIngrese los datos del proveedor:");
        System.out.print("Nombre: ");
        String nombre = scanner.nextLine();
        System.out.print("Dirección: ");
        String direccion = scanner.nextLine(); // Solicitamos la dirección
        System.out.print("Teléfono: ");
        String telefono = scanner.nextLine();

        return new Proveedor(0, nombre, direccion, telefono); // Usamos el constructor
adecuado
    }

    public int solicitarIdProveedor() {
        System.out.print("Ingrese el ID del proveedor: ");
        return scanner.nextInt();
    }

    public Proveedor solicitarDatosModificacion(Proveedor proveedor) {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nModificando proveedor ID: " + proveedor.getIdProveedor());

        System.out.print("Nuevo nombre [" + proveedor.getNombre() + "]: ");
        String nombre = scanner.nextLine();
        if (!nombre.isBlank()) proveedor.setNombre(nombre);
```

```
        System.out.print("Nueva dirección [" + proveedor.getDireccion() + "]: ");
        String direccion = scanner.nextLine();
        if (!direccion.isBlank()) proveedor.setDireccion(direccion); // Solicitamos nueva dirección

        System.out.print("Nuevo teléfono [" + proveedor.getTelefono() + "]: ");
        String telefono = scanner.nextLine();
        if (!telefono.isBlank()) proveedor.setTelefono(telefono);

        return proveedor;
    }

    public void mostrarProveedores(List<Proveedor> proveedores) {
        System.out.println("\n--- LISTADO DE PROVEEDORES ---");
        for (Proveedor proveedor : proveedores) {
            System.out.println(proveedor);
        }
    }

    public void mostrarMensaje(String mensaje) {
        System.out.println(mensaje);
    }

    public void mostrarError(String mensaje) {
        System.err.println("ERROR: " + mensaje);
    }
}
```

UTILIDADES.JAVA:

```java
package vista;

import java.util.Scanner;

public class Utilidades {

    private static Scanner scanner = new Scanner(System.in);

    public static int leerInt() {
        while (true) {
            try {
                return Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.print("Entrada inválida. Ingrese un número entero:
");
            }
        }
    }

    public static double leerDouble() {
        while (true) {
            try {
                return Double.parseDouble(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.print("Entrada inválida. Ingrese un número
decimal: ");
            }
        }
    }
}
```

VENTAVISTA.JAVA:

```
package vista;


import modelo.Venta;


import java.util.List;

import java.util.Scanner;


public class VentaVista {

    private Scanner scanner = new Scanner(System.in);


    public int mostrarMenu() {
```

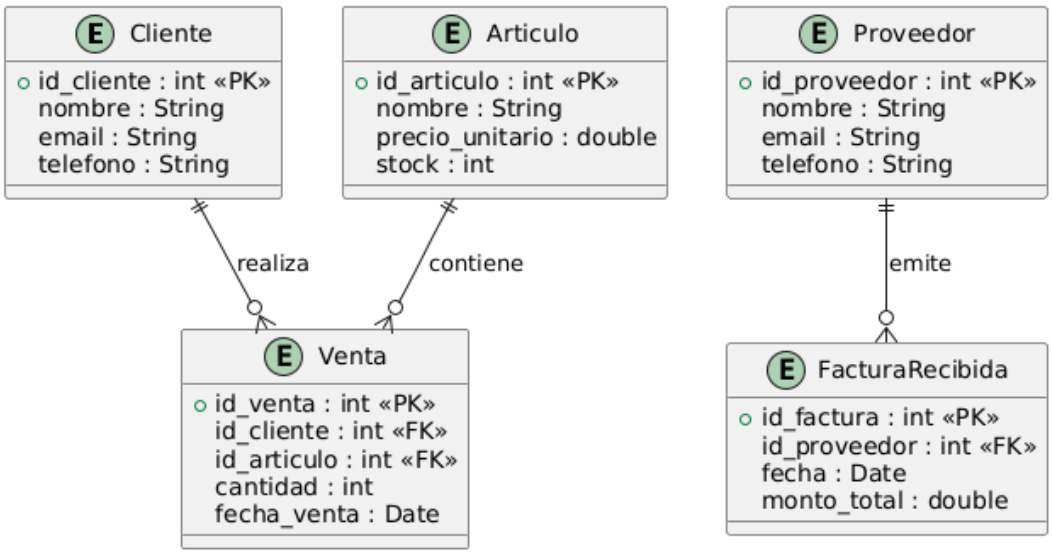
```
        System.out.println("\n--- GESTIÓN DE VENTAS ---");
        System.out.println("1. Listar ventas");
        System.out.println("2. Agregar venta");
        System.out.println("3. Modificar venta");
        System.out.println("4. Eliminar venta");
        System.out.println("0. Volver al menú principal");
        System.out.print("Seleccione una opción: ");
        return scanner.nextInt();
    }

    public Venta solicitarDatosVenta() {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nIngrese los datos de la venta:");
        System.out.print("ID Cliente: ");
        int idCliente = scanner.nextInt();
        System.out.print("ID Artículo: ");
        int idArticulo = scanner.nextInt();
        System.out.print("Cantidad: ");
        int cantidad = scanner.nextInt();
        System.out.print("Fecha de la venta (YYYY-MM-DD): ");
        String fechaVenta = scanner.next();

        return new Venta(0, idCliente, idArticulo, cantidad,
java.sql.Date.valueOf(fechaVenta));
    }

    public int solicitarIdVenta() {
        System.out.print("Ingrese el ID de la venta: ");
```

```
        return scanner.nextInt();
    }

    public Venta solicitarDatosModificacion(Venta venta) {
        scanner.nextLine(); // Limpiar buffer
        System.out.println("\nModificando venta ID: " + venta.getIdVenta());

        System.out.print("Nuevo ID Cliente [" + venta.getIdCliente() + "]: ");
        int idCliente = scanner.nextInt();
        if (idCliente != 0) venta.setIdCliente(idCliente);

        System.out.print("Nuevo ID Artículo [" + venta.getIdArticulo() + "]: ");
        int idArticulo = scanner.nextInt();
        if (idArticulo != 0) venta.setIdArticulo(idArticulo);

        System.out.print("Nueva cantidad [" + venta.getCantidad() + "]: ");
        int cantidad = scanner.nextInt();
        if (cantidad != 0) venta.setCantidad(cantidad);

        System.out.print("Nueva fecha de venta [" + venta.getFechaVenta() + "]: ");
        String fechaVenta = scanner.next();
        if (!fechaVenta.isBlank())
venta.setFechaVenta(java.sql.Date.valueOf(fechaVenta));

        return venta;
    }

    public void mostrarVentas(List<Venta> ventas) {
```

```
        System.out.println("\n--- LISTADO DE VENTAS ---");
        for (Venta venta : ventas) {
            System.out.println(venta);
        }
    }

    public void mostrarMensaje(String mensaje) {
        System.out.println(mensaje);
    }

    public void mostrarError(String mensaje) {
        System.err.println("ERROR: " + mensaje);
    }
}
```

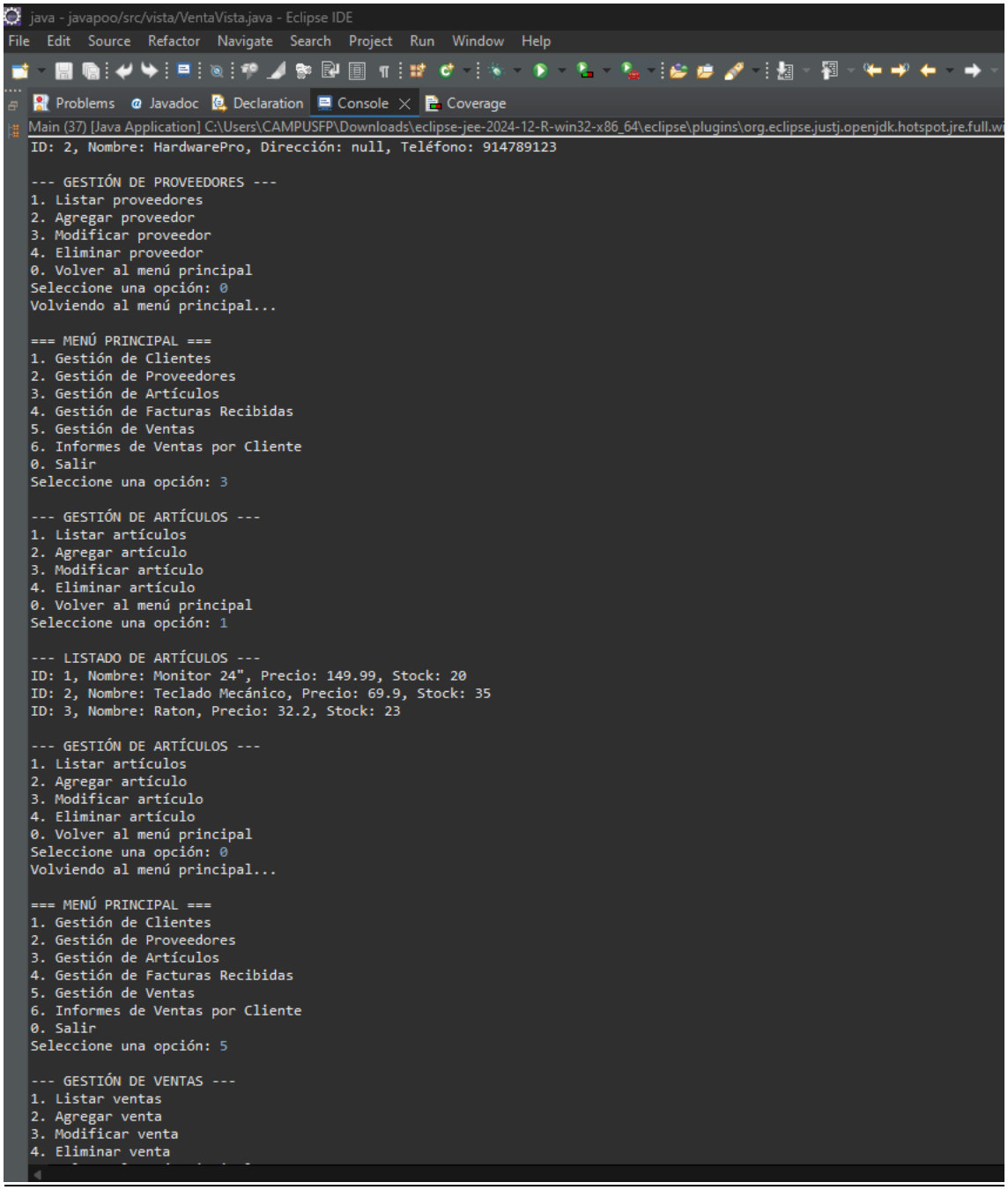
Parte 4: Actividades obligatorias

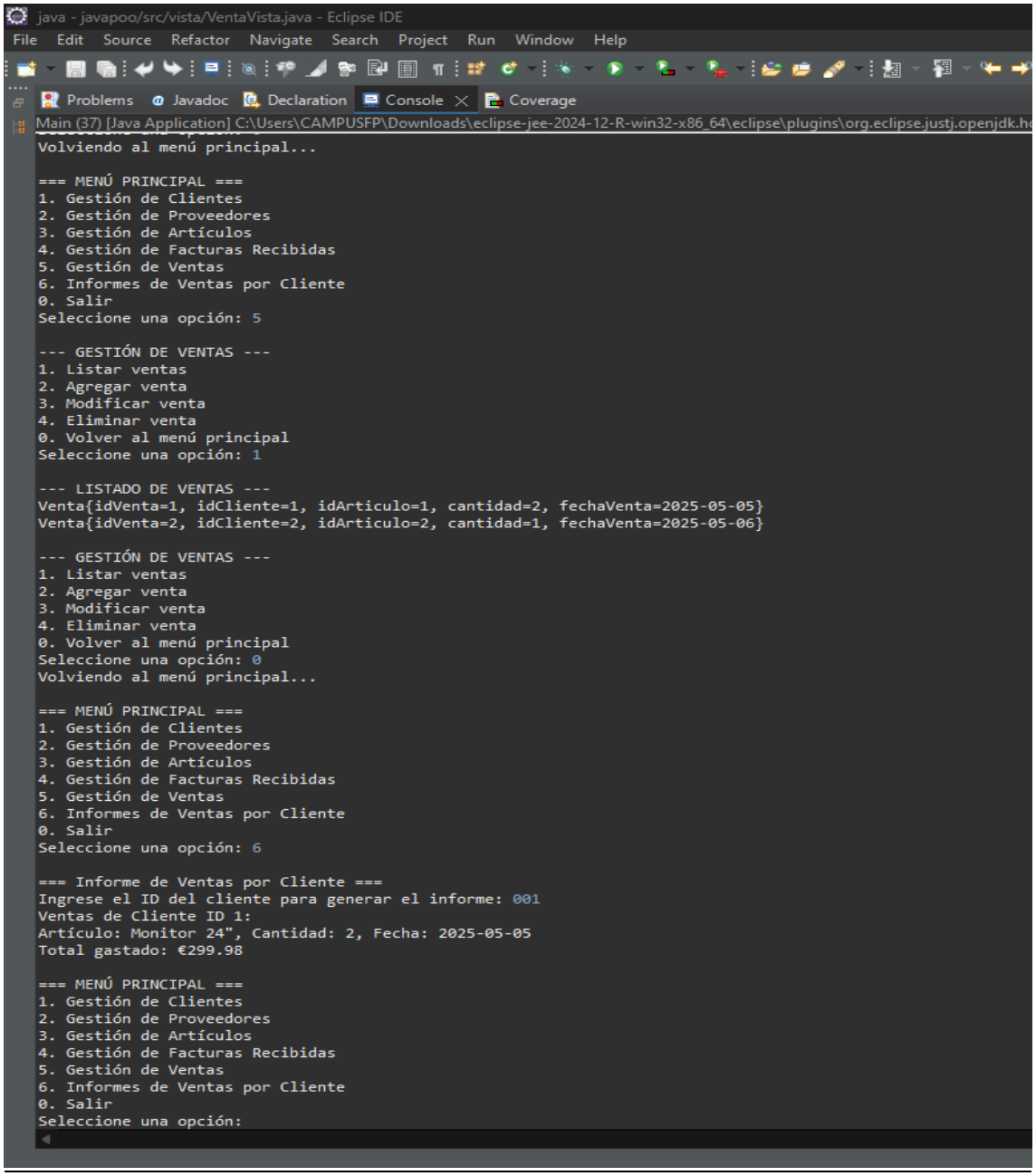
Dibujar el Diagrama de Clases de la Aplicación



FUNCIONAMIENTO DE LA APLICACIÓN:

```
=== MENÚ PRINCIPAL ===
1. Gestión de Clientes
2. Gestión de Proveedores
3. Gestión de Artículos
4. Gestión de Facturas Recibidas
5. Gestión de Ventas
6. Informes de Ventas por Cliente
0. Salir
Seleccione una opción: 1

--- Gestión de Clientes ---
1. Listar clientes
2. Agregar cliente
3. Modificar cliente
4. Eliminar cliente
0. Volver al menú principal
Seleccione una opción: 1
---- Lista de Clientes ----
ID: 1, Nombre: Juan Pérez, Email: juanp@gmail.com, Teléfono: 600123456
ID: 2, Nombre: Lucía Gómez, Email: lucia.gomez@gmail.com, Teléfono: 611987654

--- Gestión de Clientes ---
1. Listar clientes
2. Agregar cliente
3. Modificar cliente
4. Eliminar cliente
0. Volver al menú principal
Seleccione una opción: 0
Volviendo al menú principal...

=== MENÚ PRINCIPAL ===
1. Gestión de Clientes
2. Gestión de Proveedores
3. Gestión de Artículos
4. Gestión de Facturas Recibidas
5. Gestión de Ventas
6. Informes de Ventas por Cliente
0. Salir
Seleccione una opción: 2

--- GESTIÓN DE PROVEEDORES ---
1. Listar proveedores
2. Agregar proveedor
3. Modificar proveedor
4. Eliminar proveedor
0. Volver al menú principal
Seleccione una opción: 1

--- LISTADO DE PROVEEDORES ---
ID: 1, Nombre: TechDistribuciones S.A., Dirección: null, Teléfono: 913456789
ID: 2, Nombre: HardwarePro, Dirección: null, Teléfono: 914789123

--- GESTIÓN DE PROVEEDORES ---
1. Listar proveedores
2. Agregar proveedor
3. Modificar proveedor
4. Eliminar proveedor
```


java - javapoo/src/vista/VentaVista.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Coverage
Main (37) [Java Application] C:\Users\CAMPUSFP\Downloads\eclipse-jee-2024-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
ID: 2, Nombre: HardwarePro, Dirección: null, Teléfono: 914789123
--- GESTIÓN DE PROVEEDORES ---
1. Listar proveedores
2. Agregar proveedor
3. Modificar proveedor
4. Eliminar proveedor
0. Volver al menú principal
Seleccione una opción: 0
Volviendo al menú principal...
=== MENÚ PRINCIPAL ===
1. Gestión de Clientes
2. Gestión de Proveedores
3. Gestión de Artículos
4. Gestión de Facturas Recibidas
5. Gestión de Ventas
6. Informes de Ventas por Cliente
0. Salir
Seleccione una opción: 3
--- GESTIÓN DE ARTÍCULOS ---
1. Listar artículos
2. Agregar artículo
3. Modificar artículo
4. Eliminar artículo
0. Volver al menú principal
Seleccione una opción: 1
--- LISTADO DE ARTÍCULOS ---
ID: 1, Nombre: Monitor 24", Precio: 149.99, Stock: 20
ID: 2, Nombre: Teclado Mecánico, Precio: 69.9, Stock: 35
ID: 3, Nombre: Raton, Precio: 32.2, Stock: 23
--- GESTIÓN DE ARTÍCULOS ---
1. Listar artículos
2. Agregar artículo
3. Modificar artículo
4. Eliminar artículo
0. Volver al menú principal
Seleccione una opción: 0
Volviendo al menú principal...
=== MENÚ PRINCIPAL ===
1. Gestión de Clientes
2. Gestión de Proveedores
3. Gestión de Artículos
4. Gestión de Facturas Recibidas
5. Gestión de Ventas
6. Informes de Ventas por Cliente
0. Salir
Seleccione una opción: 5
--- GESTIÓN DE VENTAS ---
1. Listar ventas
2. Agregar venta
3. Modificar venta
4. Eliminar venta



CORRECIONES EN LA BASE DE DATOS