

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

SISTEMA DE UNA TIENDA AUTOMOTRIZ

**PROYECTO DEL SEGUNDO BIMESTRE PRESENTADO COMO REQUISITO
PARA APROBAR LA MATERIA DE POO**

MIGUEL ANGEL CARVAJAL SELLAN

MONICA YADIRA JAÑA GUACHI

SHAMYR SEBASTIAN QUISHPE CUADRADO

MATEO ALEJANDRO TACURI RUIZ

FREDDY ANTONIO VILLAVICENCIO ROSENDO

DIRECTOR: ING. YADIRA FRANCO

DMQ, marzo 2024

INDICE

Objetivo General.....	3
Objetivos Específicos	3
Distribución del trabajo:	3
Cronograma:	3
Roles identificados dentro del proyecto:	4
Diseño e implementación de la base de datos:	4
Desarrollo del backend:.....	5
Desarrollo del Frontend:	5
Mockups:	6
Código:.....	6
Clase BaseDatos:	6
Clase RenderImagen:	7
Clase Login:	7
Clase Admin:.....	9
Clase AddUser:	12
Clase RegistrarProductos:.....	15
Clase Carrito:	23
Clase Main:	30
Conclusiones:	30
Recomendaciones:	31
Anexos:	31
Script Base de datos:.....	31
Capturas Resultados Finales:	33

Objetivo General

- Realizar un sistema para cubrir las necesidades de una tienda de repuestos automotrices.

Objetivos Específicos

- Diseñar e implementar la estructura de la base de datos
- Realizar los mockups de la interfaz gráfica del sistema.
- Determinar los roles que requiere el sistema.

Distribución del trabajo:

Shamyr Quishpe:

- Diseño de la base de datos
- Corrección de errores en los scripts
- Seguimiento de los avances en github en función de los plazos establecidos

Mónica Jaña:

- Programación del sistema de administrados
- Control de validaciones

Mateo Tacuri:

- Diseño de las interfaces del sistema
- Revisión y análisis del ux y ui

Miguel Carvajal:

- Desarrollo del sistema de usuario
- Control de validaciones

Freddy Villavicencio:

- Generación de factura
- Encargado del documento

Cronograma:

Fecha	Shamyr	Mónica	Mateo	Miguel	Freddy
26 de febrero	Diseño de la base de datos	Programación del sistema de admin.	Diseño de las interfaces de admin	Desarrollo del sistema de usuario	Generación de factura
27 de febrero	Corrección de errores en los scripts	Control de validaciones	Revisión y análisis de UX/UI	Control de validaciones	Documentación de avances

28 de febrero	Seguimiento de avances de GitHub	Descanso	Diseño de las interfaces del Usuario	Desarrollo del sistema de usuario	Generación de factura
29 de febrero	Día libre	Día libre	Día libre	Día libre	Día libre
1 de marzo	Corrección de errores en los scripts	Control de validaciones	Programación del sistema de admin.	Control de validaciones	Descanso
2 de marzo	Seguimiento de avances de GitHub	Acabar el sistema	Documentación de avances	Acabar el sistema	Desarrollo del sistema de usuario
3 de marzo	Día libre	Día libre	Día libre	Día libre	Día libre

Roles identificados dentro del proyecto:

Dentro de la realización de nuestro proyecto se definieron dos usuarios clave.

El usuario administrador que es aquel que tiene acceso a todos los datos dentro del sistema y que puede realizar las acciones de crud y el usuario cliente que es aquel que solo tiene acceso a la información de productos para su compra y a la generación de registros de esas mismas compras.

Diseño e implementación de la base de datos:

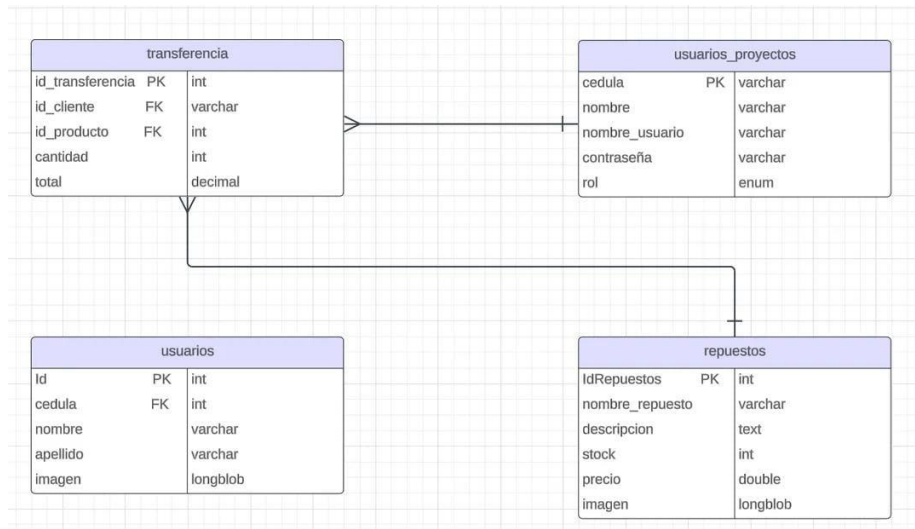
Se utilizó una base MySQL localizada en la nube para el desarrollo de nuestro sistema, a través de un estudio se definieron 3 tablas principales para el manejo de datos:

Usuario proyectos: Tabla en la que se almacenaron todos los datos de los dos tipos de usuarios que tienen acceso al sistema:

Repuestos: Tabla en la que se almacenaron todos los datos de los productos disponibles en nuestro sistema.

Transferencia: Tabla en la que almacenamos todos los datos de compra al relacionar las dos primeras tablas.

Adjuntamos el modelo lógico en la siguiente figura:



Desarrollo del backend:

Arquitectura y tecnologías: Para el desarrollo de nuestro proyecto utilizamos la plataforma de IntelliJ para poder generar el desarrollo de todo nuestro sistema, como es una herramienta bastante completa para proyectos sencillos nos bastó con esta para culminarlo.

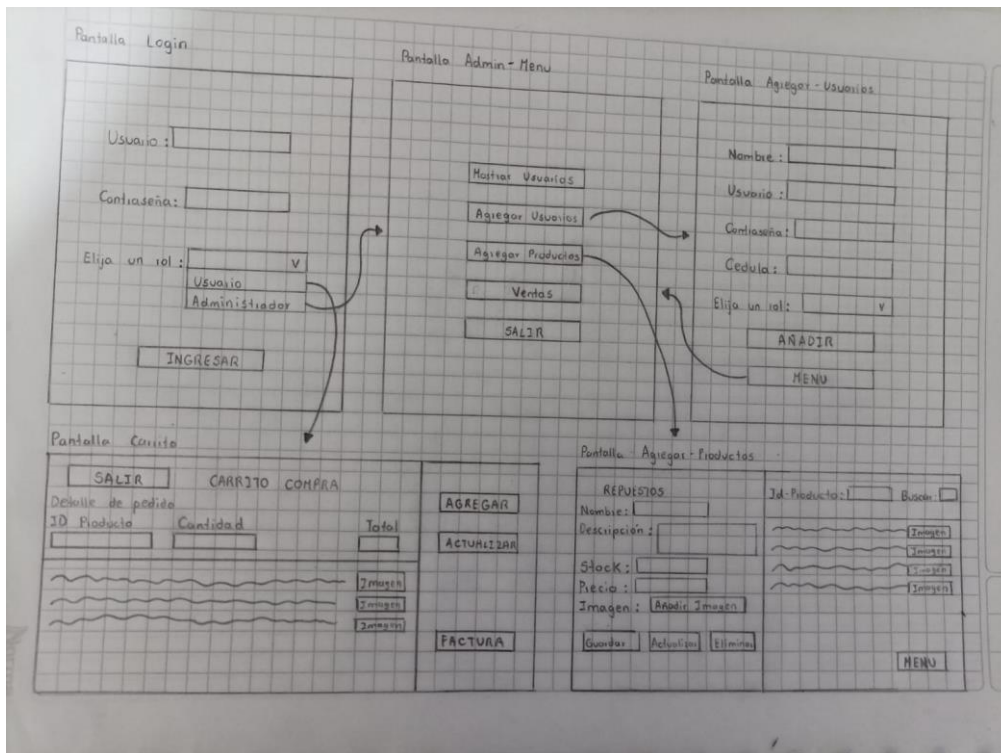
Integración de la base de datos: Se utilizó MySQL como base de datos relacional para almacenamiento de los datos del sistema. Se diseñaron los esquemas de base de datos que se consideraron más eficientes y normalizados para garantizar la integridad de los datos y la escalabilidad del sistema.

Gestión de Usuarios y Autenticación: Se implementó un sistema de gestión de usuarios que permitiera el registro e inicio de sesión de los diferentes usuarios de nuestro sistema. Además, se implementó un sistema de gestión de transacciones para procesar las compras de los usuarios, asegurando la integridad de las transacciones y la actualización del inventario.

Desarrollo del Frontend:

El desarrollo del Frontend se llevó a cabo íntegramente en el entorno de desarrollo IntelliJ, utilizando herramientas y tecnologías modernas para crear una interfaz de usuario intuitiva y atractiva para el sistema de compra de productos de la automotriz.

Mockups:



Código:

Clase BaseDatos:

En esta clase creamos la conexión a nuestra base de datos en la nube usando el controlador de conexión JDBC y el URI proporcionado por la base de datos en la nube de MYSQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class BaseDatos {
    public Connection conexionBase() {
        String url =
"jdbc:mysql://unkwsyn6m8q9ewqg:BR3VheGqPt7T416JvL7F@by4eyuiz64tfufcme0
ay-mysql.services.clever-cloud.com:3306/by4eyuiz64tfufcme0ay";
        String usuarioDB = "unkwsyn6m8q9ewqg";
        String contrasenaDB = "BR3VheGqPt7T416JvL7F";

        Connection conexion = null;

        try{
            conexion = DriverManager.getConnection(url, usuarioDB,
contrasenaDB);
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null, "Error al conectar con
la base");
        }
    }
}
```

```

        return conexion;
    }
}

```

Clase RenderImagen:

Esta clase proporciona un renderizador personalizado que muestra imágenes en una tabla Swing, el renderizado nos permite que las imágenes existentes en nuestra base de datos puedan mostrarse en una tabla esto guardándolo en un array de bytes.

```

import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.awt.image.BufferedImage;

public class RenderImagen extends DefaultTableCellRenderer {
    @Override
    public Component getTableCellRendererComponent(JTable table,
Object value, boolean isSelected, boolean hasFocus, int row, int
column) {
        if (value instanceof byte[]) {
            // Si el valor es un array de bytes (imagen)
            byte[] imageData = (byte[]) value;
            ImageIcon icon = new ImageIcon(imageData);
            Image scaledImage = icon.getImage().getScaledInstance(50,
50, Image.SCALE_SMOOTH);
            icon = new ImageIcon(scaledImage);

            setIcon(icon);
            setText(null); // Elimina el texto en la celda
        } else {
            // Si no es un array de bytes, usar el renderizador
predeterminado
            super.getTableCellRendererComponent(table, value,
isSelected, hasFocus, row, column);
        }

        return this;
    }
}

```

Clase Login:

Esta clase se encarga de buscar entre la tabla de usuarios_proyecto de nuestra base de datos y de esta manera verificar que las credenciales sean correctas y que existan en la tabla, además de esto separa el rol del usuario que este iniciando sesión entre usuario y administrador para así llevarlo a su respectiva pantalla.

```

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

```

```

import java.sql.SQLException;

public class Login extends JFrame {
    private JButton ingresarButton;
    private JTextField userText;
    private JPasswordField passField;
    private JPanel panelito;
    private JComboBox comboBox1;

    public String rol = "user";
    public Login() {
        super("LOGIN");
        setContentPane(panelito);
        ingresarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                validaciones();
            }
        });
        comboBox1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int selectedIndex = comboBox1.getSelectedIndex();
                // Realizar la acción correspondiente al rol
                seleccionado

                if (selectedIndex == 0) {
                    // Acción para Usuario
                    rol = "user";
                    System.out.println("Rol seleccionado: Usuario");
                    // Aquí puedes agregar tu lógica para el rol de
                    usuario

                } else if (selectedIndex == 1) {
                    rol = "admin";
                    // Acción para Admin
                    System.out.println("Rol seleccionado: Admin");
                    // Aquí puedes agregar tu lógica para el rol de
                    administrador

                }
            }
        });
    }

    public void validaciones() {
        String usuario = userText.getText();
        String contrasena = new String(passField.getPassword());

        if (autenticarUsuario(usuario, contrasena, rol)) {
            JOptionPane.showMessageDialog(null, "Inicio de sesion
            exitoso");
            Carrito.usuarioObtener(usuario);
            if (rol == "admin") {
                Admin vent_adm = new Admin();
                vent_adm.abrir();
                dispose();
            } else if (rol == "user") {
                Carrito carr = new Carrito();
                carr.iniciar();
                dispose();
            }
        } else {
            JOptionPane.showMessageDialog(null, "Las credenciales o el
            rol son incorrectos");
        }
    }
}

```



```

        userText.setText("");
        passField.setText("");
    }

    }

    public boolean autenticarUsuario(String nombre, String contraseña,
String rol){ //FALTARol
        BaseDatos manejadorBD = new BaseDatos();
        Connection conexion = manejadorBD.conexionBase();

        if(conexion != null){
            try{
                String sql = "SELECT * FROM usuarios_proyecto WHERE
nombre_usuario=? AND contrasena=? AND rol=?"; //FALTARol
                try(PreparedStatement stmt =
conexion.prepareStatement(sql)){
                    stmt.setString(1, nombre);
                    stmt.setString(2, contraseña);
                    stmt.setString(3, rol); //FALTARol

                    System.out.println("Consulta sql
"+stmt.toString());

                    ResultSet resultSet = stmt.executeQuery();
                    return resultSet.next();
                }
            }catch (SQLException e){
                e.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error al ejecutar
la consulta");
            }finally {
                try{
                    conexion.close();
                }catch (SQLException e){
                    e.printStackTrace();
                }
            }
        }
        return false;
    }
}

```

Clase Admin:

Esta clase es accedida siempre y cuando el rol del usuario que inicia sesión sea de administrador. Cuenta con botones que le permiten al administrador mostrar los usuarios, agregar usuarios, agregar productos, revisión de ventas. En específico en esta clase tiene 2 métodos que me muestran los usuarios y también las transferencias. Las demás opciones se ejecutan al llamar a la clase respectiva.

```

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;

```

```

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Admin extends JFrame{
    private JButton agregarUsuariosButton;
    private JButton agregarProductosButton;
    private JPanel adminPanel;
    private JButton revisionDeVentasButton;
    private JButton mostrarUsuariosButton;
    private JButton SALIRButton;

    public Admin(){
        super("PANTALLA ADMIN");
        setContentPane(adminPanel);
        mostrarUsuariosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                mostrarInformacion_tabla();

            }
        });
        agregarUsuariosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Adduser vent_adduser = new Adduser();
                vent_adduser.abrir();
                dispose();
            }
        });
        agregarProductosButton.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e) {
                RegistrarProductos registroP = new
RegistrarProductos();
                registroP.iniciar();
            }
        });
        SALIRButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Login frame = new Login();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(400,400);
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
                frame.getContentPane().setBackground(new Color(234,
211, 186));
                dispose();
            }
        });
        revisionDeVentasButton.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e) {
                mostrarInformacion_tablatra();
            }
        });
    }

    public void abrir(){

```

```

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400,400);
        setLocationRelativeTo(null);
        setVisible(true);
        this.getContentPane().setBackground(new Color(234, 211, 186));
    }

    private void mostrarInformacion_tabla() {
        BaseDatos manejadorBD = new BaseDatos();
        Connection conexion = manejadorBD.conexionBase();
        String sql = "SELECT nombre, nombre_usuario, cedula FROM usuarios_proyecto";

        try (PreparedStatement statement =
            conexion.prepareStatement(sql);
            ResultSet resultSet = statement.executeQuery()) {

            // Crear el modelo de tabla con los nombres de las
            // columnas
            DefaultTableModel tableModel = new DefaultTableModel(
                new String[]{"Nombre", "Nombre_Usuario",
                "Cedula"}, 0);

            while (resultSet.next()) {

                String nombre = resultSet.getString("Nombre");
                String nombreU =
                resultSet.getString("Nombre_Usuario");
                int id = resultSet.getInt("Cedula");

                // Agregar fila al modelo de tabla
                tableModel.addRow(new Object[]{nombre, nombreU, id});
            }

            // Crear la tabla con el modelo de datos
            JTable tabla = new JTable(tableModel);

            // Crear el panel que contendrá la tabla
            JPanel panelTabla = new JPanel(new BorderLayout());
            panelTabla.add(new JScrollPane(tabla),
            BorderLayout.CENTER);

            // Crear el marco que contendrá el panel con la tabla
            JFrame frameTabla = new JFrame("Datos en Tabla");

            frameTabla.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); //
            // Cierra solo la ventana de la tabla
            frameTabla.getContentPane().add(panelTabla);
            frameTabla.setSize(600, 400);
            frameTabla.setLocationRelativeTo(this);
            frameTabla.setVisible(true);

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error al obtener
            información de la base de datos: " + ex.getMessage(), "Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }

    private void mostrarInformacion_tablatra() {

```

```

BaseDatos manejadorBD = new BaseDatos();
Connection conexion = manejadorBD.conexionBase();
String sql = "SELECT * FROM transferencia";

try (PreparedStatement statement =
conexion.prepareStatement(sql);
    ResultSet resultSet = statement.executeQuery()) {

    DefaultTableModel tableModel = new DefaultTableModel(
        new String[]{"id_transferencia", "id_producto",
"id_cliente", "cantidad", "total"}, 0);

    while (resultSet.next()) {

        int id_t = resultSet.getInt("id_transferencia");
        int id_p = resultSet.getInt("id_producto");
        String ced = resultSet.getString("id_cliente");
        int canti = resultSet.getInt("cantidad");
        double tot = resultSet.getDouble("total");

        // Agregar fila al modelo de tabla
        tableModel.addRow(new Object[]{id_t, id_p, ced, canti,
tot});
    }

    // Crear la tabla con el modelo de datos
    JTable tabla = new JTable(tableModel);

    // Crear el panel que contendrá la tabla
    JPanel panelTabla = new JPanel(new BorderLayout());
    panelTabla.add(new JScrollPane(tabla),
BorderLayout.CENTER);

    // Crear el marco que contendrá el panel con la tabla
    JFrame frameTabla = new JFrame("Datos en Tabla");

    frameTabla.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); //
Cierra solo la ventana de la tabla
    frameTabla.getContentPane().add(panelTabla);
    frameTabla.setSize(600, 400);
    frameTabla.setLocationRelativeTo(this);
    frameTabla.setVisible(true);

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error al obtener
información de la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
}

```

Clase AddUser:

Esta clase es llamada por la clase de administrador y solicita al administrador que ingrese los datos correspondientes a la tabla para poder ingresar a la base de datos, así como también el rol respectivo del usuario ingresado.

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Adduser extends JFrame {
    private JPanel adduserPanel;
    private JButton agregarButton;
    private JButton MENUButton;
    private JTextField nombreField;
    private JTextField nombreUField;
    private JTextField paswdField;
    private JTextField cedulaField;
    private JComboBox comboBox1;

    private String rol = "user";
    public Adduser() {
        super("AGREGAR USUARIO");
        setContentPane(adduserPanel);
        agregarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ingresoFormulario();
            }
        });

        MENUButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Admin vent_admin1 = new Admin();
                vent_admin1.abrir();
                dispose();
            }
        });

        comboBox1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int selectedIndex = comboBox1.getSelectedIndex();
                // Realizar la acción correspondiente al rol
                seleccionado
                if (selectedIndex == 0) {
                    // Acción para Usuario
                    rol = "user";
                    System.out.println("Rol seleccionado: Usuario");
                    // Aquí puedes agregar tu lógica para el rol de
                    usuario
                } else if (selectedIndex == 1) {
                    rol = "admin";
                    // Acción para Admin
                    System.out.println("Rol seleccionado: Admin");
                    // Aquí puedes agregar tu lógica para el rol de
                    administrador
                }
            }
        });
    }

    public void abrir() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 400);
    }
}

```

```

        setLocationRelativeTo(null);
        setVisible(true);
        this.getContentPane().setBackground(new Color(234, 211, 186));
    }

    public void ingresoFormulario() {
        String nombre = nombreField.getText();
        String nusuario = nombreUField.getText();
        String contrsaña_reg = paswdField.getText();
        String cedula = cedulaField.getText();

        if (nombre.isEmpty() || nusuario.isEmpty() ||
        contrsaña_reg.isEmpty() || cedula.isEmpty()) {
            // Si algún campo está vacío, muestra un mensaje de error
            o realiza alguna acción adecuada
            JOptionPane.showMessageDialog(null, "Debe llenar todos los
campos");
        } else {
            // Si todos los campos están completos, llama al método
            agregarUsuarios
            agregarUsuarios(nombre, nusuario, contrsaña_reg, cedula);
        }
    }

    public void agregarUsuarios(String nombres, String usuarios,
String contra, String cedula) {
        BaseDatos manejadorBD = new BaseDatos();

        Connection conexion = manejadorBD.conexionBase();

        if (conexion != null) {
            try {
                String sql = "INSERT INTO usuarios_proyecto (nombre,
nombre_usuario, contrasena, cedula, rol) VALUES (?, ?, ?, ?, ?)";

                try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {
                    stmt.setString(1, nombres);
                    stmt.setString(2, usuarios);
                    stmt.setString(3, contra);
                    stmt.setString(4, cedula);
                    stmt.setString(5, rol);

                    stmt.executeUpdate();
                    JOptionPane.showMessageDialog(null, "Registro
Exitoso");
                }
            } catch (SQLException e) {
                JOptionPane.showMessageDialog(null, "Error al agregar
los datos");
            } finally {
                try {
                    conexion.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

Clase RegistrarProductos:

Esta clase tiene varias funcionalidades y también es llamada por la clase de admin y en ella se pueden agregar nuevos productos, actualizarlos en la tabla de repuestos, además cuenta con un apartado que muestra el producto en base a una búsqueda realizada por el id. Para poder agregar los nuevos repuestos es necesario que estén llenos todos los datos que solicita esta pantalla.

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class RegistrarProductos extends JFrame {
    private JTextField nombreP;
    private JTextField cantidadP;
    private JTextField precioP;
    private JButton subirImagenButton;
    private JButton GUARDARButton;
    private JPanel panell;
    private JTextArea descripcionP;
    private JScrollPane scrollPane;
    private JTable tableR;
    private JPanel panelDatos;
    private JPanel panelTabla;
    private JTextField buscarId;
    private JButton buscarButton;
    private JButton actualizarButton;
    private JButton eliminarButton;
    private JButton menuButton;
    private JLabel imagenLabel;

    private static byte[] imagen;

    public RegistrarProductos() {
        super("Registrar Productos");
        setContentPane(panell);

        //hacer que el texto de la descripcion se adapte al tamaño de
        la pantalla
        descripcionP.setLineWrap(true);
        descripcionP.setWrapStyleWord(true);

        scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

        //Dar tamaño al panel de los datos y al panel de la tabla
        // Establecer un tamaño específico para panelSecundario
    }
}
```

```

panelDatos.setPreferredSize(new Dimension(300, 300));
panelDatos.setBackground(new Color(234, 211, 186));
panelTabla.setPreferredSize(new Dimension(500, 400));

//Cargar los repuestos ingresados
mostrarProductos_tabla();

subirImagenButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        imagen= subirImagen(false);
    }
});
GUARDARButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        ingresoFormulario();
    }
});
menuButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Admin admin=new Admin();
        admin.abrir();
        dispose();
    }
});
actualizarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        ingresoFormulario1();
    }
});
buscarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        buscarProductos();
    }
});
eliminarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        eliminarRegistro();
    }
});
}

public void iniciar(){
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(900,600);
    setLocationRelativeTo(null);
    setVisible(true);
    this.getContentPane().setBackground(new Color(234, 211, 186));
}
public void ingresoFormulario(){
    String nombre = nombreP.getText();
    String decipcion = descripcionP.getText();
    int cantidad=0;

```



```

        double precio=0;

        try {
            cantidad =Integer.parseInt(cantidadP.getText());
        }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null,"Solo puede ingresar
numeros en cantidad");
            cantidadP.setText("");
            return;
        }

        try {
            precio = Double.parseDouble(precioP.getText());
        }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null,"Solo puede ingresar
numeros en precio");
            precioP.setText("");
            return;
        }

        System.out.println(nombre);
        System.out.println(decripcion);
        System.out.println(cantidad);
        System.out.println(precio);
        System.out.println(imagen);

        if(nombre.isEmpty()||decripcion.isEmpty()||cantidadP.getText().isEmpty()
||precioP.getText().isEmpty()||imagen==null){
            JOptionPane.showMessageDialog(null, "Todos los campos son
obligatorios");
        }else {
            try {
RegistrarProducto(nombre,decripcion,cantidad,precio,imagen);
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }
    }

    public void ingresoFormulario1(){
        String nombre = nombreP.getText();
        String decripcion = descripcionP.getText();
        int cantidad=0;
        double precio=0;

        try {
            cantidad =Integer.parseInt(cantidadP.getText());
        }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null,"Solo puede ingresar
numeros en cantidad");
            cantidadP.setText("");
            return;
        }

        try {
            precio = Double.parseDouble(precioP.getText());
        }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null,"Solo puede ingresar

```

```

    numeros en precio");
    precioP.setText("");
    return;
}

System.out.println(nombre);
System.out.println(decripcion);
System.out.println(cantidad);
System.out.println(precio);
System.out.println(imagen);

if(nombre.isEmpty() || decripcion.isEmpty() || cantidadP.getText().isEmpty() || precioP.getText().isEmpty() || imagen==null) {
    JOptionPane.showMessageDialog(null, "Todos los campos son obligatorios");
} else {
    try {
        actualizarDatos(nombre, decripcion, cantidad, precio, imagen);
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

public byte[] subirImagen(boolean exitImg) {
    if (!exitImg) {
        JFileChooser fileChooser = new JFileChooser();
        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            try {
                // Leer la imagen como un arreglo de bytes
                FileInputStream fis = new
FileInputStream(selectedFile);
                byte[] imageData = fis.readAllBytes();
                fis.close();

                // Mostrar un diálogo de confirmación
                int option = JOptionPane.showConfirmDialog(null,
"¿Desea subir esta imagen?", "Confirmar", JOptionPane.YES_NO_OPTION);
                if (option == JOptionPane.YES_OPTION) {
                    // Obtener direccion de la imagen
                    JOptionPane.showMessageDialog(null, "La imagen
se subió correctamente" );
                    return imageData;
                } else {
                    JOptionPane.showMessageDialog(null, "La imagen
no se subió");
                }
            } catch (IOException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error al
subir la imagen: " + ex.getMessage());
            }
        }
        return null;
    }
    return null;
}
}

```

```

        private void RegistrarProducto(String nombre, String descripcion,
int cantidad, double precio,byte[] imageData) throws SQLException {

        BaseDatos manejadorBD = new BaseDatos();
        Connection conexion = manejadorBD.conexionBase();

        if(conexion != null) {
            try {
                String sql = "INSERT INTO repuestos
(nombreRepuesto,descripcion,stock,precio,imagen) VALUES (?,?, ?,
?,?)";

                try(PreparedStatement stmt =
conexion.prepareStatement(sql)) {

                    stmt.setString(1,nombre);
                    stmt.setString(2,descripcion);
                    stmt.setInt(3,cantidad);
                    stmt.setDouble(4,precio);
                    stmt.setBytes(5, imageData);
                    stmt.executeUpdate();
                    JOptionPane.showMessageDialog(null, "Registro
Exitoso");

                    nombreP.setText("");
                    descripcionP.setText("");
                    cantidadP.setText("");
                    precioP.setText("");
                    imagen=new byte[0];

                }
            }catch (SQLException e){
                JOptionPane.showMessageDialog(null, "Error al agregar
los datos");
            } finally {
                try{
                    conexion.close();
                }catch (SQLException e){
                    e.printStackTrace();
                }
            }
        }

        private void mostrarProductos_tabla() {
            BaseDatos manejadorBD = new BaseDatos();
            Connection conexion = manejadorBD.conexionBase();
            String sql = "SELECT * FROM repuestos";

            try (PreparedStatement statement =
conexion.prepareStatement(sql);
                ResultSet resultSet = statement.executeQuery()) {

                // Crear el modelo de tabla con los nombres de las
columnas
                DefaultTableModel tableModel = new DefaultTableModel(new
Object[][]{},
                    new String[]{"Id", "Nombre",
"Descripcion","Stock","Precio","Imagen"});

                //Limpiar la tabla
                for (int i = 0; i < tableR.getRowCount(); i++) {
                    tableModel.removeRow(i);
                    i -= 1;
                }
            }
        }
    }
}

```

```

    }

    while (resultSet.next()) {
        int id=resultSet.getInt("IdRepuestos");
        String nombre = resultSet.getString("nombreRepuesto");
        String descripcion =
resultSet.getString("descripcion");
        int stock = resultSet.getInt("stock");
        double precio= resultSet.getDouble("precio");
        byte[] imagen=resultSet.getBytes("imagen");

        // Agregar fila al modelo de tabla
        tableModel.addRow(new
Object[]{id,nombre,descripcion,stock,precio,imagen});
    }

    tableR.setModel(tableModel);

    // Aplicar el renderizador personalizado a la columna de
imágenes
    tableR.getColumnModel().getColumn(5).setCellRenderer(new
RenderImagen());

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error al obtener
información de la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }

}

public void buscarProductos(){
    BaseDatos manejadorBD = new BaseDatos();
    Connection conexion = manejadorBD.conexionBase();
    int id= Integer.parseInt(buscarId.getText());

    if (conexion != null) {
        try {
            // Preparar la consulta SQL para verificar la
autenticación
            String sql = "SELECT *FROM repuestos Where
idRepuestos=? ";
            try (PreparedStatement pstmt =
conexion.prepareStatement(sql)) {
                // Establecer los valores de los parámetros en la
consulta
                pstmt.setInt(1, id);
            // Ejecutar la consulta para obtener el resultado
            System.out.println("Consulta SQL: " +
pstmt.toString()); // Imprimir la consulta SQL
            ResultSet resultSet = pstmt.executeQuery();

            // Procesar el resultado del ResultSet
            while (resultSet.next()) {
                int idProducto =
resultSet.getInt("IdRepuestos");
                String nombre =
resultSet.getString("nombreRepuesto");
                String descripcion =

```

```

resultSet.getString("descripcion");
        int stock = resultSet.getInt("stock");
        double precio = resultSet.getDouble("precio");
        byte[] imagen=resultSet.getBytes("imagen");

        // Mostrar los en la interfaz gráfica
        nombreP.setText(nombre);
        descripcionP.setText(descripcion);
        cantidadP.setText(String.valueOf(stock));
        precioP.setText(String.valueOf(precio));

        if (imagen != null) {
            // Crear un ImageIcon a partir de los
datos de la imagen
            ImageIcon imageIcon = new
ImageIcon(imagen);

            // Escalar la imagen al tamaño de un
carnet (5.5cm x 3.5cm)
            Image image = imageIcon.getImage();
            Image scaledImage =
image.getScaledInstance(165, 150, Image.SCALE_SMOOTH);
            ImageIcon scaledImageIcon = new
ImageIcon(scaledImage);

            // Establecer el ImageIcon escalado en el
JLabel
            imagenLabel.setIcon(scaledImageIcon);
        } else {
            JOptionPane.showMessageDialog(this, "El
usuario no tiene una imagen asociada.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
        //System.out.println("ID: " + idProducto + ",
Nombre: " + nombre + ", Descripción: " + descripcion + ", Stock: " +
stock + ", Precio: " + precio+", Imagen: "+imagen);
    }

}

} catch (SQLException e) {
    e.printStackTrace(); // Imprimir el seguimiento de la
pila para diagnóstico
    JOptionPane.showMessageDialog(null, "Error al ejecutar
la consulta: " + e.getMessage());
} finally {
    try {
        conexion.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}

}

private void actualizarDatos(String nombre, String descripcion,
int cantidad, double precio,byte[] imageData) throws SQLException {
    BaseDatos manejadorBD = new BaseDatos();
    int id= Integer.parseInt(buscarId.getText());

    try (Connection conexion = manejadorBD.conexionBase());

```

```

        PreparedStatement statement =
conexion.prepareStatement("UPDATE repuestos SET nombreRepuesto = ?,
descripcion = ?, stock = ?, precio=?, imagen=? WHERE idRepuestos =
?")) {

        // Establecer los valores de los parámetros en el
PreparedStatement
        statement.setString(1, nombre);
        statement.setString(2, descripcion);
        statement.setInt(3, cantidad);
        statement.setDouble(4, precio);
        statement.setBytes(5, imageData);
        statement.setInt(6, id);

        // Ejecutar la actualización
int filasActualizadas = statement.executeUpdate();

        // Verificar si se realizaron actualizaciones y mostrar un
mensaje correspondiente
        if (filasActualizadas > 0) {
            JOptionPane.showMessageDialog(this, "Datos
actualizados correctamente.");
        } else {
            JOptionPane.showMessageDialog(this, "No se encontró el
producto con ID " + id, "Error", JOptionPane.ERROR_MESSAGE);
        }

    } catch (SQLException | NumberFormatException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error al actualizar
datos en la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void eliminarRegistro() {
    BaseDatos manejadorBD = new BaseDatos();
    int id= Integer.parseInt(buscarId.getText());
    try (Connection conexion = manejadorBD.conexionBase();
        PreparedStatement statement =
conexion.prepareStatement("DELETE FROM repuestos WHERE idRepuestos =
?")) {

        statement.setInt(1, id);

        int filasEliminadas = statement.executeUpdate();

        if (filasEliminadas > 0) {
            JOptionPane.showMessageDialog(this, "Producto
eliminado correctamente.");
        } else {
            JOptionPane.showMessageDialog(this, "No se encontró el
producto con ID " + id, "Error", JOptionPane.ERROR_MESSAGE);
        }

    } catch (SQLException | NumberFormatException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error al eliminar
registro en la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

```

```
}  
}
```

Clase Carrito:

Esta clase es llamada cuando en el login se detecta el rol como usuario, en esta clase se muestran los productos disponibles al usuario, el usuario ingresa el id del producto que desea comprar y la cantidad de producto que desea y lo agrega al carrito de esta manera también agrega en la tabla de transferencias y también actualiza el stock disponible en la tabla de repuestos, hay un botón que permite actualizar la tabla que se muestra al cliente. Y también usando la librería itext se genera una factura en formato pdf cuando el usuario de click en generar reporte.

```
import javax.swing.*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
import com.itextpdf.text.Document;  
import com.itextpdf.text.DocumentException;  
import com.itextpdf.text.pdf.PdfPTable;  
import com.itextpdf.text.pdf.PdfWriter;  
public class Carrito extends JFrame {  
    private JButton regresarButton;  
    private JButton agregarProductoButton;  
    private JTextField cantidadField;  
    private JTable table1;  
    private JLabel mostrarTotal;  
    private JLabel mostrarPrecio;  
    private JLabel mostrartotal2;  
    private JPanel panelcarrito;  
    private JTextField idField;  
    private JButton actualizarButton;  
    private JButton generarReporteButton;  
    static int cantidad = 0;  
    static int id_fijo = 0;  
    static double precio = 0.00;  
    static double precio_final = 0.00;  
  
    static String usuarioActual = "";  
  
    static String cedulaUsuario = "";  
  
    public Carrito() {  
        super("Compra");  
        setContentPane(panelcarrito);  
  
        mostrarProductos_tabla();  
        regresarButton.addActionListener(new ActionListener() {
```

```

        @Override
        public void actionPerformed(ActionEvent e) {

        }

    });
    regresarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Login frame = new Login();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(400, 400);
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
            frame.getContentPane().setBackground(new Color(234,
211, 186));
            dispose();
        }
    });
    agregarProductoButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            validaciones();
        }
    });
    actualizarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarProductos_tabla();
        }
    });
    generarReporteButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Document documento= new Document();

            try{
                String ruta= System.getProperty("user.home");
                PdfWriter.getInstance(documento,new
FileOutputStream(ruta+"/OneDrive/Escritorio/Reporte_compra.pdf"));

                System.out.println("Estoy aqui");
                documento.open();
                PdfPTable tabla= new PdfPTable(5);
                tabla.addCell("id_transferencia");
                tabla.addCell("id_producto");
                tabla.addCell("id_cliente");
                tabla.addCell("cantidad");
                tabla.addCell("total");
                System.out.println("Estoy aqui2");
                BaseDatos manejadorBD = new BaseDatos();
                Connection conexion = manejadorBD.conexionBase();
                String sql = "SELECT * FROM transferencia";

                try (PreparedStatement statement =
conexion.prepareStatement(sql);
                    ResultSet resultSet =
statement.executeQuery()) {

                    while (resultSet.next()) {

```



```

        tabla.addCell(resultSet.getString(1));
        tabla.addCell(resultSet.getString(2));
        tabla.addCell(resultSet.getString(3));
        tabla.addCell(resultSet.getString(4));
        tabla.addCell(resultSet.getString(5));
        System.out.println("Estoy aqui3");
    }
    documento.add(tabla);
} catch (SQLException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error al
obtener información de la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
}
documento.close();
JOptionPane.showMessageDialog(null, "Reporte
Generado");
} catch (DocumentException | HeadlessException |
FileNotFoundException exe) {}

    });
}

public void iniciar() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(900, 600);
    setLocationRelativeTo(null);
    setVisible(true);
    this.getContentPane().setBackground(new Color(234, 211, 186));
    cedulaObtener();
}

private void mostrarProductos_tabla() {
    BaseDatos manejadorBD = new BaseDatos();
    Connection conexion = manejadorBD.conexionBase();
    String sql = "SELECT * FROM repuestos";

    try (PreparedStatement statement =
conexion.prepareStatement(sql);
        ResultSet resultSet = statement.executeQuery()) {

        DefaultTableModel tableModel =
crearModeloTabla(resultSet);

        table1.setModel(tableModel);

        // Aplicar el renderizador personalizado a la columna de
imágenes
        table1.getColumnModel().getColumn(5).setCellRenderer(new
RenderImagen());

    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error al obtener
información de la base de datos: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
}

```

```

        private DefaultTableModel crearModeloTabla(ResultSet resultSet)
        throws SQLException {
            DefaultTableModel tableModel = new DefaultTableModel(new
            Object[][] {},
                new String[] {"Id", "Nombre",
                "Descripcion", "Stock", "Precio", "Imagen"});

            //Limpiar la tabla
            while (tableModel.getRowCount() > 0) {
                tableModel.removeRow(tableModel.getRowCount() - 1);
            }

            while (resultSet.next()) {
                int id=resultSet.getInt("IdRepuestos");
                String nombre = resultSet.getString("nombreRepuesto");
                String descripcion = resultSet.getString("descripcion");
                int stock = resultSet.getInt("stock");
                double precio= resultSet.getDouble("precio");
                byte[] imagen=resultSet.getBytes("imagen");

                // Agregar fila al modelo de tabla
                tableModel.addRow(new
                Object[] {id,nombre,descripcion,stock,precio,imagen});
            }

            return tableModel;
        }

        public void validaciones() {
            String idTexto = idField.getText().trim();
            String cantidadTexto = cantidadField.getText().trim();

            // Verificar si alguno de los campos está vacío
            if (idTexto.isEmpty() || cantidadTexto.isEmpty()) {
                JOptionPane.showMessageDialog(null, "No pueden haber
                campos vacíos", "Error", JOptionPane.ERROR_MESSAGE);
                idField.setText("");
                cantidadField.setText("");
                return; // Salir del método si hay campos vacíos
            }

            id_fijo = Integer.parseInt(idField.getText());
            int cantidadRecibida =
            Integer.parseInt(cantidadField.getText());

            if (autenticarProducto(id_fijo)) {
                cantidadActual();
                if (cantidadRecibida > cantidad) {
                    JOptionPane.showMessageDialog(null, "El id es
                    correcto, pero la cantidad excede el stock disponible");
                } else {
                    JOptionPane.showMessageDialog(null, "El id es
                    correcto, producto agregado al carrito");
                    precio_final = (cantidadRecibida * precio);
                    mostrarTotal.setText(String.valueOf(precio_final));
                    insertarDatos(cantidadRecibida);
                    actualizarRepuesto(cantidadRecibida);
                    mostrarProductos_tabla();
                }
            } else {
                JOptionPane.showMessageDialog(null, "El id es

```

```

incorrecto");
        idField.setText("");
        cantidadField.setText("");
    }
}

public boolean autenticarProducto(int id) { //FALTARol
    BaseDatos manejadorBD = new BaseDatos();
    Connection conexion = manejadorBD.conexionBase();

    if (conexion != null) {
        try {
            String sql = "SELECT * FROM repuestos WHERE
IdRepuestos=?"; //FALTARol
            try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {
                stmt.setInt(1, id);

                System.out.println("Consulta sql " +
stmt.toString());

                ResultSet resultSet = stmt.executeQuery();
                return resultSet.next();

            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error al ejecutar
la consulta");
        } finally {
            try {
                conexion.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    return false;
}

public static void cantidadActual() {
    BaseDatos manejadorBD = new BaseDatos();

    // Obtener la conexión a la base de datos
    try (Connection conexion = manejadorBD.conexionBase()) {
        // Verificar si la conexión es nula
        if (conexion == null) {
            JOptionPane.showMessageDialog(null, "No se pudo
establecer la conexión a la base de datos", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Salir del método si no se pudo establecer
la conexión
        }

        // Preparar la consulta SQL para obtener todos los
usuarios
        String sql = "SELECT stock, precio FROM repuestos WHERE
IdRepuestos=?";
        try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {

```

```

        // Ejecutar la consulta para obtener el resultado
        stmt.setInt(1, id_fijo);

        ResultSet resultSet = stmt.executeQuery();

        if (resultSet.next()) {
            cantidad = resultSet.getInt("stock");
            System.out.println(cantidad);
            precio = resultSet.getDouble("precio");
            System.out.println(precio);
        }
    } catch (SQLException e) {
        // Capturar cualquier excepción SQL
        JOptionPane.showMessageDialog(null, "Error al recuperar
usuarios: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        e.printStackTrace(); // Imprimir la traza de la pila para
diagnóstico
    }
}

public static void usuarioObtener(String usuario) {
    usuarioActual = usuario;
}

public static void cedulaObtener() {
    BaseDatos manejadorBD = new BaseDatos();

    // Obtener la conexión a la base de datos
    try (Connection conexion = manejadorBD.conexionBase()) {
        // Verificar si la conexión es nula
        if (conexion == null) {
            JOptionPane.showMessageDialog(null, "No se pudo
establecer la conexión a la base de datos", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Salir del método si no se pudo establecer
la conexión
        }

        // Preparar la consulta SQL para obtener todos los
usuarios
        String sql = "SELECT cedula FROM usuarios_proyecto WHERE
nombre_usuario=?";
        try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {
            // Ejecutar la consulta para obtener el resultado
            stmt.setString(1, usuarioActual);

            ResultSet resultSet = stmt.executeQuery();

            if (resultSet.next()) {
                cedulaUsuario = resultSet.getString("cedula");
                System.out.println(cedulaUsuario);
            }
        }
    } catch (SQLException e) {
        // Capturar cualquier excepción SQL
        JOptionPane.showMessageDialog(null, "Error al recuperar
usuarios: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        e.printStackTrace(); // Imprimir la traza de la pila para
diagnóstico
    }
}

```

```

    }
}

public static void insertarDatos(int cantidad1) {
    BaseDatos manejadorBD = new BaseDatos();

    Connection conexion = manejadorBD.conexionBase();

    if (conexion != null) {
        try {
            String sql = "INSERT INTO transferencia (id_producto,
id_cliente, cantidad, total) VALUES (?, ?, ?, ?)";

            try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {
                stmt.setInt(1, id_fijo);
                stmt.setString(2, cedulaUsuario);
                stmt.setInt(3, cantidad1);
                stmt.setDouble(4, precio_final);

                stmt.executeUpdate();
                JOptionPane.showMessageDialog(null, "Registro
Exitoso");
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error al agregar
la transaccion");
        } finally {
            try {
                conexion.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

public static void actualizarRepuesto(int cantidad1) {
    BaseDatos manejadorBD = new BaseDatos();

    // Obtener la conexión a la base de datos
    try (Connection conexion = manejadorBD.conexionBase()) {
        // Verificar si la conexión es nula
        if (conexion == null) {
            JOptionPane.showMessageDialog(null, "No se pudo
establecer la conexión a la base de datos", "Error",
JOptionPane.ERROR_MESSAGE);
            return; // Salir del método si no se pudo establecer
la conexión
        }

        // Preparar la consulta SQL para obtener todos los
usuarios
        String sql = "UPDATE repuestos SET stock = ? WHERE
IdRepuestos = ?";
        try (PreparedStatement stmt =
conexion.prepareStatement(sql)) {
            int nuevoCantidad = cantidad - cantidad1;
            // Ejecutar la consulta para obtener el resultado
            stmt.setInt(1, nuevoCantidad);
            stmt.setInt(2, id_fijo);

```

```

        stmt.executeUpdate();

        JOptionPane.showMessageDialog(null, "Actualizacion de
cantidad");
    }
} catch (SQLException e) {
    // Capturar cualquier excepción SQL
    JOptionPane.showMessageDialog(null, "Error al recuperar
usuarios: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    e.printStackTrace(); // Imprimir la traza de la pila para
diagnóstico
}
}
}

```

Clase Main:

En el creamos el frame principal donde se mostrará de inicio la pantalla del login.

```

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

public class Main {
    public static void main(String[] args) {
        Login frame = new Login();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,400);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
        frame.getContentPane().setBackground(new Color(234, 211,
186));
    }
}

```

Conclusiones:

Eficiencia en el desarrollo: La programación orientada a objetos permitió un desarrollo eficiente del sistema de compra de productos para la automotriz, gracias a la reutilización de código y la estructura modular.

Flexibilidad y escalabilidad: La estructura orientada a objetos del sistema garantiza su flexibilidad y escalabilidad, permitiendo adaptarse fácilmente a cambios en los requisitos del negocio y la incorporación de nuevas funcionalidades en el futuro.

Separación de roles: La implementación de roles de administrador y usuario facilita una gestión eficaz del sistema, mejorando la seguridad y el control de acceso.

Interfaz de usuario intuitiva: Se priorizó una interfaz de usuario intuitiva y fácil de usar para mejorar la usabilidad y la experiencia del usuario, tanto para administradores como para usuarios finales.

Pruebas exhaustivas: Se realizaron pruebas unitarias, de integración y de aceptación exhaustivas para garantizar la funcionalidad y la integridad del sistema antes de su implementación.

Recomendaciones:

Mantenimiento continuo: Se recomienda realizar actualizaciones y mantenimiento continuo del sistema para garantizar su óptimo funcionamiento a lo largo del tiempo.

Feedback del usuario: Recopilar y analizar el feedback de los usuarios puede proporcionar información valiosa para futuras mejoras del sistema.

Seguridad: Es fundamental implementar medidas de seguridad robustas para proteger los datos sensibles de clientes y transacciones financieras.

Optimización del rendimiento: Monitorear y optimizar el rendimiento del sistema es esencial para garantizar tiempos de respuesta rápidos y una experiencia de usuario fluida.

Capacitación del personal: Proporcionar capacitación adecuada al personal que utilizará el sistema maximiza su eficacia y minimiza los errores.

En resumen, el proyecto fue exitoso gracias a la eficiencia en el desarrollo, la flexibilidad del sistema y una interfaz de usuario intuitiva. Para asegurar su éxito continuo, se recomienda mantener un enfoque en el mantenimiento, la seguridad, la optimización del rendimiento y la capacitación del personal.

Anexos:

Script Base de datos:

```
CREATE TABLE `repuestos` (  
  
  `IdRepuestos` int NOT NULL AUTO_INCREMENT,  
  
  `nombreRepuesto` varchar(302) NOT NULL,  
  
  `descripcion` tinytext NOT NULL,  
  
  `stock` int NOT NULL,  
  
  `precio` double NOT NULL,  
  
  `imagen` longblob NOT NULL,  
  
  PRIMARY KEY (`IdRepuestos`)
```

```

) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;

CREATE TABLE `transferencia` (

  `id_transferencia` int NOT NULL AUTO_INCREMENT,

  `id_producto` int DEFAULT NULL,

  `id_cliente` varchar(20) DEFAULT NULL,

  `cantidad` int DEFAULT NULL,

  `total` decimal(10,2) DEFAULT NULL,

  PRIMARY KEY (`id_transferencia`),

  KEY `id_producto` (`id_producto`),

  KEY `id_cliente` (`id_cliente`),

  CONSTRAINT `transferencia_ibfk_1` FOREIGN KEY (`id_producto`) REFERENCES
`repuestos` (`IdRepuestos`),

  CONSTRAINT `transferencia_ibfk_2` FOREIGN KEY (`id_cliente`) REFERENCES
`usuarios_proyecto` (`cedula`)

) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8;

CREATE TABLE `usuarios_proyecto` (

  `cedula` varchar(20) NOT NULL,

  `nombre` varchar(100) NOT NULL,

  `nombre_usuario` varchar(50) DEFAULT NULL,

  `contrasena` varchar(255) NOT NULL,

  `rol` enum('admin','user') DEFAULT 'user',

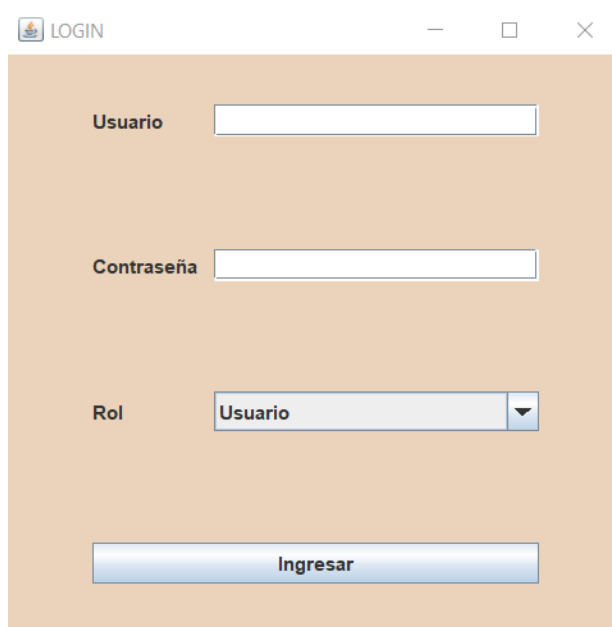
  PRIMARY KEY (`cedula`),

  UNIQUE KEY `nombre_usuario` (`nombre_usuario`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

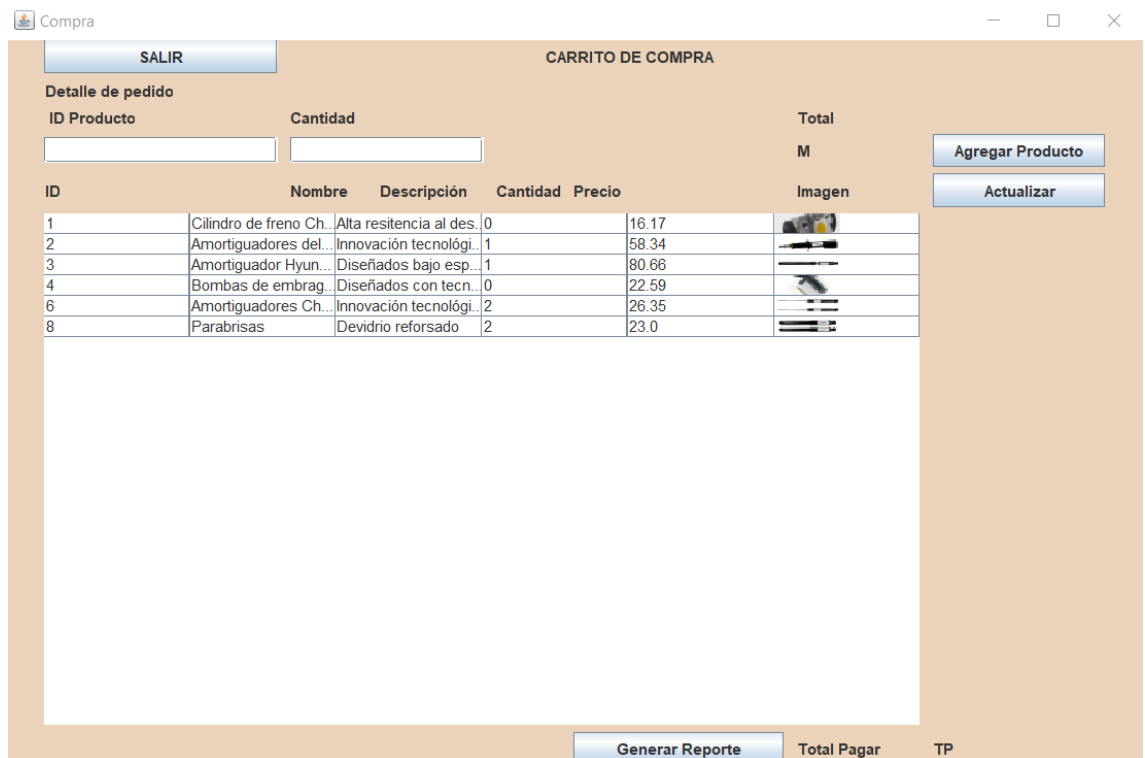
```


Capturas Resultados Finales:









A screenshot of a web application window titled "LOGIN". The window has a light brown background. It contains three input fields: "Usuario" (Username), "Contraseña" (Password), and "Rol" (Role). The "Rol" field is a dropdown menu with "Usuario" selected. Below these fields is a blue button labeled "Ingresar" (Login).

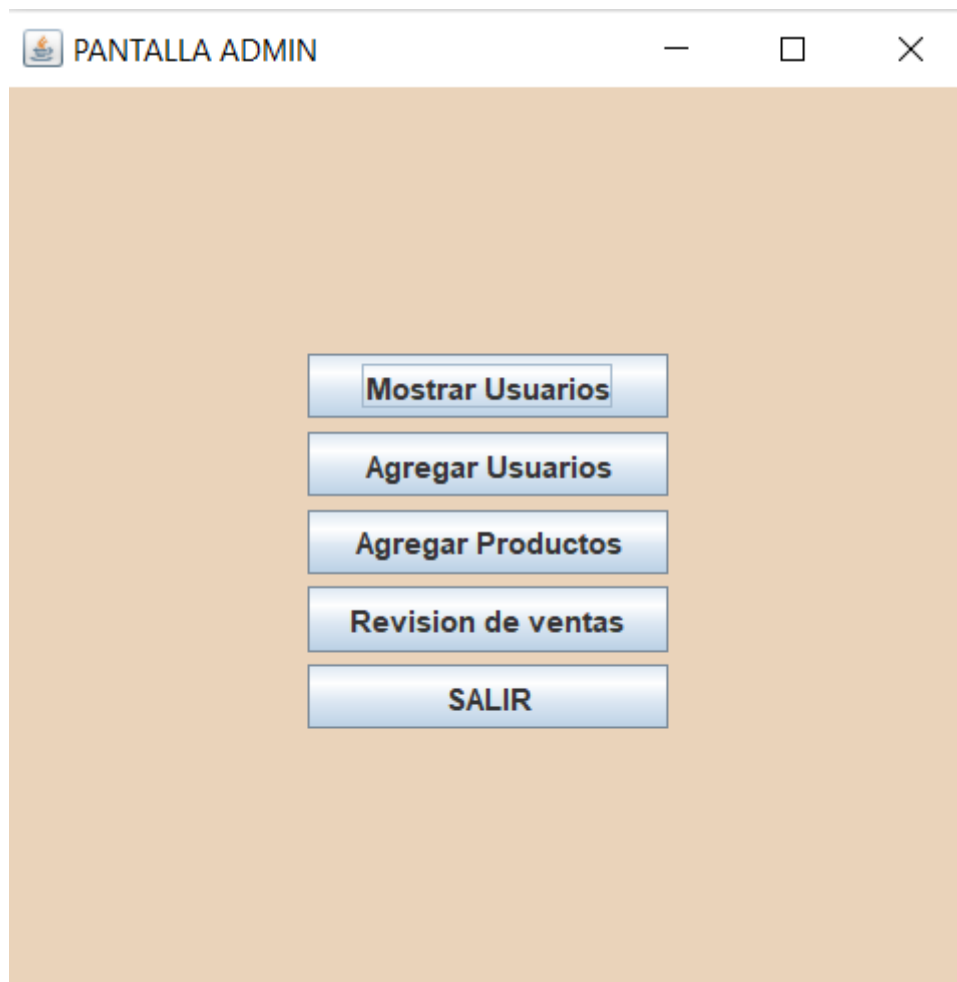
Pantalla Login



A screenshot of a web application window titled "CARRITO DE COMPRA" (Shopping Cart). The window has a light brown background. At the top left is a blue button labeled "SALIR" (Exit). Below it is a section titled "Detalle de pedido" (Order Detail) with two input fields: "ID Producto" and "Cantidad". To the right of these fields is a "Total" label and a blue button labeled "Agregar Producto" (Add Product). Below the input fields is a table with the following columns: "ID", "Nombre", "Descripción", "Cantidad", "Precio", and "Imagen". The table contains 8 rows of product data. To the right of the table is a blue button labeled "Actualizar" (Update). At the bottom of the window is a blue button labeled "Generar Reporte" (Generate Report) and two labels: "Total Pagar" and "TP".

ID	Nombre	Descripción	Cantidad	Precio	Imagen
1	Cilindro de freno Ch...	Alta resistencia al des...	0	16.17	
2	Amortiguadores del...	Innovación tecnológi...	1	58.34	
3	Amortiguador Hyun...	Diseñados bajo esp...	1	80.66	
4	Bombas de embrag...	Diseñados con tecn...	0	22.59	
6	Amortiguadores Ch...	Innovación tecnológi...	2	26.35	
8	Parabrisas	Devidrio reforzado	2	23.0	

Pantalla Carrito




Pantalla Admin

The screenshot shows a window titled "Datos en Tabla" with a standard Windows-style title bar. It contains a table with three columns: "Nombre", "Nombre_Usuario", and "Cedula". The table has 8 rows of data. Below the table is a large, empty light gray rectangular area.

Nombre	Nombre_Usuario	Cedula
dsfsd		0
Mateo Bernal	MateoB	1233214567
Pepe Guardiola	PepeG	1234567123
Juan Perez	JuanP	1234567890
Shamyr Quishpe	ShamyrQ	1726082827
Miguel Carvajal	MiguelC	1726864000
Mateo	MateoT	1752876209

Pantalla Mostrar Usuario


AGREGAR USUARIO
—
□
×

Nombre

Nombre de Usuario

Contraseña


Cedula

Rol Usuario ▼

Agregar

MENU

Pantalla Agregar Usuario


Registrar Productos
—
□
×

REPUESTOS

Nombre:

Descripcion:







Cantidad:

Precio:

Imagen: Subir Imagen


GUARDAR Actualizar Eliminar

Ingrese el Id del producto: Buscar

1	Cilindro de fre...	Alta resistencia ...	0	16.17	
2	Amortiguadore...	Innovación tec...	1	58.34	
3	Amortiguador ...	Diseñados baj...	1	80.66	
4	Bombas de e...	Diseñados con...	0	22.59	
6	Amortiguadore...	Innovación tec...	2	26.35	
8	Parabrisas	Devidrio refors...	2	23.0	

Menu

Pantalla Agregar Productos

 Datos en Tabla

id_transferencia	id_producto	id_cliente	cantidad	total
1	1	1726864000	7	32.0
2	1	1726864000	2	32.0
3	1	1726864000	2	32.0
4	1	1726864000	2	32.0
5	1	1726864000	2	32.0
6	1	1726864000	2	32.0
7	3	1726864000	3	241.0
8	3	1726864000	1	80.0
9	1	1726864000	3	48.0
10	4	1726864000	4	90.0
11	4	1726864000	2	45.0
12	6	1726864000	1	26.0
13	2	1726864000	1	58.0
14	2	1726864000	2	116.0
15	2	1726864000	1	58.0
16	6	1726864000	1	26.0
17	6	1726864000	1	26.0
18	4	1726864000	1	22.0
19	4	1726864000	1	22.0
20	4	1726864000	2	45.0
21	3	1233214567	1	80.0

Pantalla de revisión de ventas



FACTURA GENERADA

 Cliente: MateoT
 Cedula de identidad: 1752876209

Producto adquirido: Amortiguadores Chevrolet Aveo/Posterior
 Cantidad: 1
 Precio final: \$26.35

Generación de factura