Reporte de Práctica sobre Punto de Venta

DIEGO RAÚL HERNÁNDEZ CARDEÑA
DAVID MORALES MENDIETA
HÉCTOR ADRIÁN PAULO VÁZQUEZ
DANIEL YOSEF SANTIAGO GARCÍA
UNIVERSIDAD LA SALLE OAXACA
INGENIERÍA EN SOFTWARE Y SISTEMAS COMPUTACIONALES

Resumen - Este reporte de práctica aborda el estudio y la implementación de interfaces gráficas en el marco de la asignatura de Laboratorio de Programación Orientada a Objetos. Se explora el funcionamiento del programa detrás de un punto de venta y la interfaz del usuario (UX) que ponen en perspectiva su aplicación en la vida real.

I. INTRODUCCION

Se presenta la documentación completa del sistema de punto de venta desarrollado por el equipo de ingeniería en software. El sistema proporciona una solución integral para la gestión de ventas y el control de inventario, con una interfaz gráfica de usuario desarrollada en Java. Este documento tiene como objetivo proporcionar una comprensión detallada de la implementación del sistema, desde su diseño hasta su funcionamiento en producción.

II. FUNDAMENTO TEÓRICO

A. *Principios de Ingeniería de Software*

La ingeniería de software es el proceso sistemático de diseño, desarrollo, operación y mantenimiento de sistemas de software. Este proceso se basa en principios fundamentales que incluyen la modularidad, la reutilización de código, la cohesión y el acoplamiento adecuados, la abstracción y la encapsulación. Estos principios guían el diseño del sistema de punto de venta, asegurando un código limpio, mantenible y escalable.

B. *Programación Orientada a Objetos (POO)*

La programación orientada a objetos es un paradigma de programación que se basa en la creación y manipulación de objetos que interactúan entre sí. En el proyecto de punto de venta, se utilizan conceptos de POO como clases, objetos, herencia, encapsulamiento y polimorfismo para modelar entidades del mundo real, como productos, clientes y transacciones. Esta metodología de desarrollo facilita la modularidad y la reutilización del código, así como la organización y la estructuración del sistema.

C. *Conexión a Bases de Datos*

Las bases de datos son componentes esenciales en el diseño y la implementación de sistemas de software eficientes. En el proyecto de punto de venta, se hizo uso de una base de datos para gestionar la información que se ingresa y se manipula dentro del programa desde la inserción de los datos del usuario que el programa requiere para loguearlo, también para lograr la manipulación de los datos en cuestión de almacenamiento de productos y transacciones.

D. *Desarrollo de Interfaces de Usuario*

El desarrollo de interfaces de usuario se centra en el diseño y la implementación de interfaces gráficas intuitivas y atractivas para los usuarios. En el proyecto de punto de venta, se utiliza Java Swing, una biblioteca de componentes gráficos para Java, para desarrollar la interfaz de usuario. Se aplican principios de diseño de interfaces de usuario, como la consistencia, la retroalimentación y la simplicidad, para garantizar una experiencia de usuario satisfactoria.

E. Patrones de Diseño

Los patrones de diseño son soluciones probadas para problemas comunes de diseño de software. En el proyecto de punto de venta, se aplican varios patrones de diseño, como el patrón MVC (Modelo-Vista-Controlador) para separar la lógica de negocio de la interfaz de usuario, y el patrón Observador para notificar a los componentes de la interfaz sobre cambios en los datos. Estos patrones promueven la modularidad, la flexibilidad y la extensibilidad del sistema.

F. Gestión de Datos

La gestión de datos se refiere al almacenamiento, recuperación y manipulación de información en el sistema. En el proyecto de punto de venta, se utilizan sistemas de gestión de bases de datos relacionales (RDBMS) para almacenar datos de productos, clientes, transacciones, etc. Se aplican técnicas de normalización de bases de datos y consultas SQL para garantizar la integridad y la consistencia de los datos.

G. Seguridad de la Información

La seguridad de la información es fundamental en cualquier sistema de software que maneje datos sensibles. En el proyecto de punto de venta, se implementan medidas de seguridad como cifrado de datos para proteger la confidencialidad, integridad y disponibilidad de la información, sin embargo, el sistema sigue siendo vulnerable a inyecciones SQL.

III. METODOLOGÍA

A. Interfaz del proyecto

1. Página de inicio de sesión (Fig. 1):

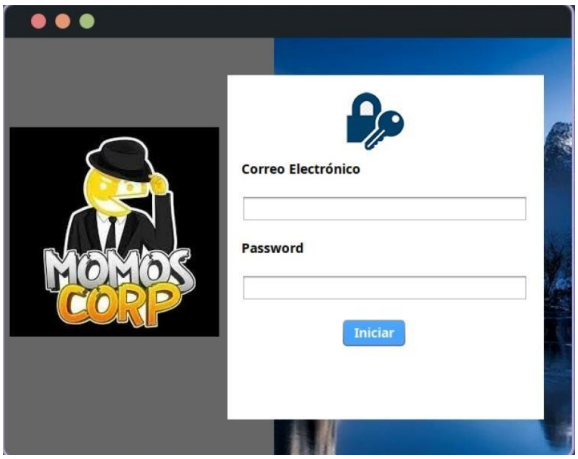

Figura 1. Página de inicio de sesión

Esta página tiene la tarea de restringir el acceso al usuario a no ser que ingrese sus datos en motivo de validar sus credenciales y brindarle los servicios que ofrece la página del punto de venta de ‘Momos Corp’.

B. Generar Venta.

1. Página tab2 (Fig. 13): Explicación de la interfaz

Figura 2. Página tab2

A esta pestaña se le es adjudicada la tarea de recopilar los datos necesarios para generar la venta de un producto, tomando en cuenta la cantidad de piezas del producto que el comprador requiera y la comparación con el stock que se le puede ofrecer.

C. Datos del Cliente

1. Página de inserción de los datos del comprador. (Fig. 3)

Figura 3. Página de inserción de datos del cliente

En esta sección es importante que el usuario ingrese sus datos con una cautela importante, pues de lo contrario la base de datos tendrá un registro erróneo de él y esto puede crear futuros problemas.

Internamente, esta página cumple con las funciones de buscar clientes, ingresar clientes, borrar y modificar clientes.

D. Datos del Proveedor.

1. Página de inserción de los datos del proveedor. (Figura 4)

Figura 4. Página de inserción de los datos del proveedor.

La interfaz cuenta con una pestaña (tab3) en la que según el orden de los TextFields, estos tienen que ser rellenados con los datos del proveedor.
Internamente, el programa cuenta con las funcionalidades para borrar el registro de un proveedor, ingresar uno, buscar y modificar registros de proveedores según sea necesario.

E. Datos del Producto.

1. Pestaña de inserción de los datos de los productos. (Figura 5)

Figura 5. Página de inserción de datos por producto.

Tab4 es la pestaña encargada de conectar la información real que describe la identidad de cada producto, ingresada por el usuario, con la base de datos que la almacenará y la gestionará.
Internamente el programa cuenta con la funcionalidad de exportar los datos de los productos a una hoja nueva en Microsoft Excel, también es posible ingresar, modificar, buscar y borrar el registro de cada producto de la base de datos.

F. Información de Ventas.

1. Pestaña de inserción de datos de ventas. Figura 6.

Figura 6. Página de inserción de los datos de las ventas

Esta pestaña se encarga de mostrar TextFields en donde se requiera ingresar los datos de las ventas que se llevarán a cabo en el punto de venta.

G. Datos de la Empresa.

1. Pestaña de inserción de los datos de la empresa. (Figura 7)

Figura 7. Página de inserción de datos de la empresa.

Tab 6 es la pestaña con la tarea de comunicar a la base de datos los datos que el usuario ingrese, los cuales, en este caso describirían a la empresa en razón de mantener un control fiscal mediante el control de movimientos que se llevan a cabo tras la actualización en la base de datos después de cada venta.

IV. CODIFICACIÓN

A. Conexión a base de datos

Para la conexión a base de datos se utilizó mysql y mariadb (dependiendo el S.O), utilizando el acrónimo CRUD para realizar operaciones en la base de datos

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexion {
    Connection con;
    public Connection getConnection(){
        try {
            //mariadb, 3307, sistemaventas ---> windows
            //mysql, 3306, sistemaventa ---> Ubuntu
            String myBD = "jdbc:mariadb://localhost:3307/sistemaventas?serverTimezone=UTC";
            con = DriverManager.getConnection(myBD, "root","");
            return con;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

B. Logueo

Para el logueo se realizó una consulta a la base de datos buscando si el correo ingresado existe, de ser así toda la información retornada en la consulta es guardada en un objeto Usuario, para posteriormente verificar la contraseña ingresada con la contraseña encriptada en la BDD

```java
public Usuario user (String correo, String pass) {
        Usuario usuario = new Usuario();
        String sql = "SELECT * FROM users WHERE mail = ?";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setString(1,correo);
            rs = ps.executeQuery();

            if (rs.next()) {
                usuario.setId_user(rs.getInt("id_user"));
                usuario.setName_c(rs.getString("name_c"));
                usuario.setTipo_user(rs.getInt("tipo_user"));
                usuario.setMail(rs.getString("mail"));
                usuario.setPasswordS(rs.getString("passwordS"));
            }
        } catch (SQLException e) {
            // TODO: handle exception
            System.out.println(e.toString());
        }
        return usuario;
    }
```

```java
public void validar () {
        String correo = textFieldCorreoElectronico.getText();
        String pass = String.valueOf(passwordFieldContraseña1.getPassword());

        Encriptacion encriptacion = new Encriptacion();
        String passEncriptada = encriptacion.encryptPassword(pass);


        if (!"".equals(correo) || !"".equals(pass)) {
            user = verifLog.user(correo, passEncriptada);
            if (user.getMail()!=null && user.getPasswordS()!=null) {
                if (encriptacion.checkPassword(pass, user.getPasswordS())) {
                    System.out.println("EXITO");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Correo o contraseña incorrecta");
            }
        }
    }
```

C. Listar Objetos

Se utilizó un array para recorrer cada objeto, almacenando la información guardada en la base de datos

```java
public void ListarCliente() {
        List<Cliente> ListarCl = client.listarCliente();
        modelo = (DefaultTableModel) tableCliente.getModel();
        Object[] ob = new Object[6];
        for (int i = 0; i < ListarCl.size(); i++) {
            ob[0] = ListarCl.get(i).getId();
            ob[1] = ListarCl.get(i).getDni();
            ob[2] = ListarCl.get(i).getNombre();
            ob[3] = ListarCl.get(i).getTelefono();
            ob[4] = ListarCl.get(i).getDireccion();
            ob[5] = ListarCl.get(i).getRazon();
            modelo.addRow(ob);
        }
        tableCliente.setModel(modelo);
    }
```

```java
public List listarCliente(){
    List<Cliente> ListaCl = new ArrayList();
    String sql = "SELECT * FROM clientes";
    try {
        con = cn.getConnection();
        if (con == null) {
            return ListaCl;
        }
        ps = con.prepareStatement(sql);
        rs = ps.executeQuery();
        while(rs.next()){
            Cliente cl = new Cliente();
            cl.setId(rs.getInt("id"));
            cl.setDni(rs.getInt("dni"));
            cl.setNombre(rs.getString("nombre"));
            cl.setTelefono(rs.getString("telefono"));
            cl.setDireccion(rs.getString("direccion"));
            cl.setRazon(rs.getString("razon"));
            ListaCl.add(cl);
        }
    } catch (SQLException e) {
        System.out.println(e.toString());
    }
    return ListaCl;
}
```

D. Eliminar Objetos

Para eliminar un objeto, se realiza un delete en la base de datos donde el id sea igual al mismo del objeto seleccionado

```java
public boolean EliminarProveedor(int id){
        String sql = "DELETE FROM proveedor WHERE id = ?";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setInt(1,id);
            ps.execute();
            return true;
        } catch (SQLException e) {
            // TODO: handle exception
            System.out.println(e.toString());
            return false;
        } finally{
            try {
                con.close();
            } catch (Exception e) {
                System.out.println(e.toString());
            }
        }
    }
```

```java
private void btnEliminarProveedorActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (!"".equals(txtIdProveedor.getText())){
            int pregunta = JOptionPane.showConfirmDialog(null,"Estas seguro de eliminar?");
            if (pregunta == 0){
                int id = Integer.parseInt(txtIdProveedor.getText());
                prDao.EliminarProveedor(id);
                LimpiarTable();
                ListarProveedor();
                LimpiarProveedor();
                cbxProveedorProducto.removeAllItems();
                prDao.ConsultarProveedor(cbxProveedorProducto);

            }
        }
    }
```

E. Registrar Objetos

Para registrar objetos, se realiza un INSERT en la base de datos, guardando todos los valores ingresados, así mismo se validan los campos ingresados para evitar errores al ingresar los atributos de cada objeto

```
public boolean RegistrarProveedor (Proveedor pr){
        String sql = "INSERT INTO proveedor (ruc, nombre, telefono, direccion, razon) VALUES (?,?,?,?,?)";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setInt(1, pr.getRuc());
            ps.setString(2,pr.getNombre());
            ps.setString(3,pr.getTelefono());
            ps.setString(4,pr.getDireccion());
            ps.setString(5,pr.getRazon());
            ps.execute();
            return true;
        } catch (SQLException e) {
            // TODO: handle exception
            System.out.println(e.toString());
            return false;
        } finally{
            try {
                con.close();
            } catch (SQLException e) {
                System.out.println(e.toString());
            }
        }
    }
```

```
private void btnGuardarProveedorActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnGuardarProveedorActionPerformed
        // TODO add your handling code here:
        boolean avanzar = true;
        if (!"".equals(txtCodigoProducto.getText()) || !"".equals(txtDescripcionProducto.getText()) || !"".equals(txtCantidadProducto.getText()) ||
            !"".equals(cbxProveedorProducto.getSelectedItem())|| !"".equals(txtPrecioProducto.getText())){
            String cantidadText = txtCantidadProducto.getText();
            if (!cantidadText.isEmpty()){
                if (cantidadText.matches("\\d+")){
                    int cantidad = Integer.parseInt(cantidadText);
                    pro.setStock(cantidad);
                } else {
                    JOptionPane.showMessageDialog(null,"El valor de Stock no es un entero");
                    avanzar = false;
                }
            }
            String PrecioText = txtPrecioProducto.getText();
            if (!PrecioText.isEmpty()){
                if (PrecioText.matches("\\d+(\\.\\d+)?")){
                    double Precio = Double.parseDouble(PrecioText);
                    pro.setPrecio(Precio);
                } else {
                    JOptionPane.showMessageDialog(null,"El valor del precio no es correcto");
                    avanzar = false;
                }
            }

            if (avanzar == true){
                pro.setCodigo(txtCodigoProducto.getText());
                pro.setNombre(txtDescripcionProducto.getText());
                //pro.setStock(Integer.parseInt(txtCantidadProducto.getText()));
                pro.setProveedor(cbxProveedorProducto.getSelectedItem().toString());
                //pro.setPrecio(Double.parseDouble(txtPrecioProducto.getText()));
                prdDao.RegistrarProductos(pro);
                JOptionPane.showMessageDialog(null,"Producto registrado");
                LimpiarTable();
                LimpiarProducto();
                ListarProductos();
            }
        } else {
            JOptionPane.showMessageDialog(null,"Los campos estan vacios");
        }
    }//GEN-LAST:event_btnGuardarProveedorActionPerformed
```

F. Actualizar Objetos

Para actualizar los atributos de algún objeto, se almacena toda la información en un objeto nuevo, y se realiza un UPDATE en la base de datos, donde el ID sea igual al mismo del objeto a modificar

```
public boolean ModificarProveedor(Proveedor pr){
        String sql = "UPDATE proveedor SET ruc = ?, nombre = ?, telefono = ?, direccion = ?, razon = ? WHERE id = ?";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(sql);
            ps.setInt(1, pr.getRuc());
            ps.setString(2, pr.getNombre());
            ps.setString(3,pr.getTelefono());
            ps.setString(4,pr.getDireccion());
            ps.setString(5, pr.getRazon());
            ps.setInt(6,pr.getId());
            ps.execute();
            return true;
        } catch (SQLException e) {
            // TODO: handle exception
            System.out.println(e.toString());
            return false;
        } finally {
            try {
                con.close();
            } catch (SQLException e) {
                // TODO: handle exception
                System.out.println(e.toString());
            }
        }
    }
```

```
private void btnActualizarProveedorActionPerformed(java.awt.event.ActionEvent evt) {// GEN-FIRST:event_btnActualizarProveedorActionPerformed
    // TODO add your handling code here:
        boolean avanzar = true;
        if ("".equals(txtIdProveedor.getText())){
            JOptionPane.showMessageDialog(null,"Seleccione una fila");
        } else {
            if (!"".equals(txtRucProveedor.getText()) || !"".equals(txtNombreProveedor.getText()) || !"".equals(txtTelefonoProveedor.getText()) ||
                !"".equals(txtDireccionProveedor.getText()) || !"".equals(txtRazonProveedor.getText())){
                String telText = txtTelefonoProveedor.getText();
                // Verifica si el campo de teléfono está vacío
                if (!telText.isEmpty()) {
                    // Verifica si el valor ingresado es un número
                    if (telText.matches("\\d+")) {
                        // Si es un número, establece el valor del teléfono
                        pr.setTelefono(telText);
                    } else {
                        // Si no es un número, muestra un mensaje de error
                        JOptionPane.showMessageDialog(null,"El valor del Teléfono no es un número valido: " + telText);
                        avanzar = false;
                    }
                }
                if (avanzar) {
                    // Verificar y establecer los otros valores si todo está bien con el teléfono
                    pr.setRuc(Integer.parseInt(txtRucProveedor.getText()));
                    pr.setNombre(txtNombreProveedor.getText());
                    pr.setDireccion(txtDireccionProveedor.getText());
                    pr.setRazon(txtRazonProveedor.getText());
                    pr.setId(Integer.parseInt(txtIdProveedor.getText()));
                    prDao.ModificarProveedor(pr);
                    JOptionPane.showMessageDialog(null,"Proveedor modificado");
                    LimpiarTable();
                    ListarProveedor();
                    LimpiarProveedor();
                    cbxProveedorProducto.removeAllItems(); // Elimina todos los elementos actuales
                    prDao.ConsultarProveedor(cbxProveedorProducto); // Consulta y agrega los proveedores nuevamente

                }
            }
        }
    }// GEN-LAST:event_btnActualizarProveedorActionPerformed
```

G. Generar una venta

Para generar una venta, se ocuparon 4 principales funciones: Registrar venta, Registrar detalles, Actualizar el stock y generar un PDF de la información de la venta

```
private void btnGenerarVentaActionPerformed(java.awt.event.ActionEvent evt) {// GEN-FIRST:event_btnGenerarVentaActionPerformed
        // TODO add your handling code here:
        RegistrarVenta();
        registrarDetalle();
        actulizarStock();
        pdf();
        limpiarTableVenta();
        limpiarClienteVenta();
    }// GEN-LAST:event_btnGenerarVentaActionPerformed
```

```
public int RegistrarVenta (Venta v) {
        String SQL = "INSERT INTO ventas (cliente, vendedor, total) VALUES (?,?,?)";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(SQL);
            ps.setString(1, v.getCliente());
            ps.setString(2, v.getVendedor());
            ps.setDouble(3, v.getTotal());
            ps.execute();

        } catch (SQLException e) {
            // TODO: handle exception
            System.out.println(e.toString());
        } finally {
            try {
                con.close();
            } catch (SQLException e) {
                // TODO: handle exception
                System.out.println(e.toString());
            }
        }
        return r;
    }

    public int registrarDetalle (Detalle dV) {
        String SQL = "INSERT INTO detalle (cod_producto, cantidad, precio, id_venta) VALUES (?,?,?,?)";
        try {
            con = cn.getConnection();
            ps = con.prepareStatement(SQL);
            ps.setString(1, dV.getCodProd());
            ps.setInt(2, dV.getCantidad());
            ps.setDouble(3, dV.getPrecio());
            ps.setInt(4, dV.getId());
            ps.execute();
        } catch (SQLException e) {
            System.out.println(e.toString());
        } finally {
            try {
                con.close();
            } catch (SQLException e) {
                // TODO: handle exception
                System.out.println(e.toString());
            }
        }

        return r;
    }
```

```
private void RegistrarVenta () {
        String cliente = txtNombreClienteVenta.getText();
        String vendedor = labelVendedor.getText();
        double monto = totalaPagar;
        v.setCliente(cliente);
        v.setVendedor(vendedor);
        v.setTotal(monto);
        vDAO.RegistrarVenta(v);
    }

    private void registrarDetalle() {
        int id = vDAO.idVenta();
        for (int i = 0; i < tableVenta.getRowCount(); i++) {
            String cod = tableVenta.getValueAt(i, 0).toString();
            int cant = Integer.parseInt(tableVenta.getValueAt(i,2).toString());
            double precio = Double.parseDouble(tableVenta.getValueAt(i, 3).toString());

            dV.setCodProd(cod);
            dV.setCantidad(cant);
            dV.setPrecio(precio);
            dV.setId(id);
            vDAO.registrarDetalle(dV);
        }
    }
```

```
private void actulizarStock () {
        for (int i = 0; i < tableVenta.getRowCount(); i++) {
            String cod = tableVenta.getValueAt(i, 0).toString();
            int cant = Integer.parseInt(tableVenta.getValueAt(i, 2).toString());
            pro = proDao.BuscarPro(cod);
            int stockActual = pro.getStock() - cant;
            vDAO.actualizarStock(stockActual, cod);
        }
    }
```

V. CONCLUSIONES

El proyecto de punto de venta de 'Momos Corp' se basa en una estructura bien organizada que asegura una gestión eficiente de los datos y la seguridad de la información.

Gestión de Datos

El sistema utiliza sistemas de gestión de bases de datos relacionales (RDBMS) y técnicas de normalización para garantizar la integridad y consistencia de los datos almacenados, como productos, clientes y transacciones. Las consultas SQL permiten una recuperación y manipulación eficiente de esta información.

Seguridad de la Información

Se han implementado medidas de seguridad como el cifrado de datos para proteger la confidencialidad e integridad de la información. Sin embargo, el sistema es vulnerable a inyecciones SQL, lo que resalta la necesidad de mejorar las prácticas de seguridad.

Interfaz del Proyecto

La interfaz incluye varias funcionalidades clave:

Página de Inicio de Sesión: Restringe el acceso validando las credenciales del usuario.
Generar Venta: Recopila los datos necesarios para procesar ventas, verificando la disponibilidad del stock.
Datos del Cliente: Facilita la inserción, búsqueda, modificación y eliminación de datos del cliente, asegurando la precisión de los registros en la base de datos.

REFERENCIAS

[1] IEEE. (2023). "IEEE Editorial Style Manual

Autores

Daniel Yosef Santiago García
Héctor Adrián Paulo Vásquez
Diego Raúl Hernández Cardeña
David Morales Mendieta