Universidad de San Carlos de Guatemala
Brayan Oswaldo Rojas Morales
Carnet: 202108307
Ing

Manual Técnico

Seccion: G

Guatemala, 15 de marzo de 2023

Introducción

El propósito de este manual técnico es explicar detalladamente el proceso de realización del programa llamado: "USAC-Delivery", explicar cada una de las partes del código que lo compone.

Este manual esta dirigido a todos lo programadores que estén interesados en entender la manera en que se desarrollo el programa, que lenguaje se usó y la lógica del mismo.

Arquitectura de software

El programa descrito en este manual se desarrollo pensando que que hay dos tipos de usuario que van a poder ingresar al sistema, el primero es un usuario administrador y luego tres tipos de usuario distintos que primero van a tener que registrarse en un apartado de registro, el administrador va a poder administrar los siguientes módulos:

- Manejo de kioskos
- Manejo de regiones y precios
- Manejo de departamentos y municipios
- Reportes

Y segundo, un usuario que en el momento que se logee va a poder elegir ser un usuario individual, usuario empresarial, o Kiosko, y los tres tipos de usuario van a poder manejar los siguientes módulos:

- Registro de tarjeta de crédito/débito
- Registro de datos de facturación
- Cotización de paquetes
- Compra
- Descargar factura y guía (etiqueta de envío)
- Ver envíos solicitados

Especificaciones Técnicas

Lenguaje usado para desarrollar el programa: Java

IDE: Apache Netbeans IDE 17

Framework: Jcalendar.

En este programa se hizo el uso de la memoria temporal para guardar los datos del programa, es decir que no se usó ningún tipo de bases de datos para almacenar la información.

Bibliotecas principales:

- javax.swing.JOptionPane
- java.awt.event.ItemEvent
- import java.awt.event.ItemListener
- import javax.swing.JFrame
- javax.swing
- java.util.regex.Matcher;
- import java.util.regex.Pattern;

Se hizo uso de listas en html para mostrar la información importante del cliente con el motivo de que el cliente pudiera ver de una mejor y mas ordenada manera de ver la información deseada

Documentación del código fuente

Módulos registro y Loguin

En el módulo de Loguin, primero se instancio las clases necesarias para que al momentos de que se inicie sesión puedan cargar todos los datos necesarios de todo el programa en general.

```
public class Inicio_sesion extends javax.swing.JFrame {

    Registro rg = new Registro();
    Principal pc = new Principal();
    administrador_principal admin = new administrador_principal();

    usuario usr[] = new usuario[10];
    Kioskos kiosko[] = new Kioskos[10];
    Admin_Departamentos depa[] = new Admin_Departamentos[10];
    Admin_Municipios muni[] = new Admin_Municipios[10];
    Tarjetas_debito tar[] = new Tarjetas_debito[10];
    Datos_facturacion fac[] = new Datos_facturacion[10];
```

Luego se agregaron métodos que toman como argumentos un arreglo de objetos, que permiten actualizar las variables correspondientes de la clase que las contiene con los nuevos arreglos

```
public void actualizarUsuario(usuario usr[]) {
    this.usr = usr;
}

public void actualizarKiosko(Kioskos kiosko[]) {
    this.kiosko = kiosko;
}

public void actualizarDepartamento(Admin_Departamentos depa[]) {
    this.depa = depa;
}

public void actualizarMunicipios(Admin_Municipios muni[]) {
    this.muni = muni;
}

public void actualizarTarjetas(Tarjetas_debito tar[]) {
    this.tar = tar;
}
```

Luego se usa el constructor de la clase para cargar las clases correspondientes con las variables

```
public Inicio_sesion() {
    initComponents();
    rg.cargarClase(this, usr, kiosko);
    pc.cargarClase(this, tar, fac, depa, muni);
    admin.cargarClase(this, kiosko, depa, muni);
}
```

Posteriormente, el usuario procedera a ingresar los datos correspondientes en dos textbox como el correo y la contraseña, primero se procedio a verificar que las dos texto box no estuvieran vacias de la siguiente manera:

```
if (txtCorreoIni.getText().equals("") || txtContraIni.getText().equals("")) {
    JOptionPane.showMessageDialog(null, "Error, usuario o correo vacios");
} else {
```

los datos del administrador han sido quemados en el mismo programa de la siguiente manera

```
if (txtCorreoIni.getText().equals("ipc1_202108307@ipc1delivery.com") && txtContraIni.getText().equals("202108307")) {
    existeAdmin = true;
}
```

y para verificar que los datos del usuario ya esten ingresados se uso el siguiente código:

```
boolean existe = false;
boolean existeAdmin = false;
for (int i = 0; i < usr.length; i++) {
    if (usr[i] != null) {
        if (usr[i].getCorreo().equals(txtCorreoIni.getText()) && usr[i].getContraseña().equals(txtContraIni.getText())) {
            existe = true;
            pc.setUsr(usr[i]);

            break;
        }
    }
}
```

```
if (txtCorreoIni.getText().equals("ipc1_202108307@ipc1delivery.com") && txtContraIni.getText().equals("202108307")) {
    existeAdmin = true;
}
```

```
    }
    if (existe) {
        this.setVisible(false);
        pc.setVisible(true);
        txtCorreoIni.setText("");
        txtContraIni.setText("");

    } else if (existeAdmin) {
        this.setVisible(false);
        admin.setVisible(true);
    } else {

        JOptionPane.showMessageDialog(null, "Usuario o correo incorrecta");
    }
}
```

Modulo de registro

Bibliotecas usadas en este modulo:

```
package rpg2;

import java.awt.Color;
import javax.swing.JOptionPane;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

Primero se declararon las variables de tipo arreglo que se van a usar en la clase

```
usuario[] usr;
Inicio_sesion is;
Kioskos[] kiosko;
```

Se creo una modulo para cargar los datos de los arreglos en las variables declaradas

```
public void cargarClase(Inicio_sesion is, usuario[] usr, Kioskos[] kiosko) {
    this.is = is;
    this.usr = usr;
    this.kiosko = kiosko;
}
```

Luego el usuario tiene que proceder a llenar los datos correspondientes que se le piden, luego se hacen algunas validaciones como por ejemplo que las cajas donde se ingresan los datos no esten vacias.

```
private void btnRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtCorreo.getText().equals("") || txtNombre2.getText().equals("") || txtApellido.getText().equals("") || txtContraseña
        JOptionPane.showMessageDialog(null, "Hay algun dato que no ha sido ingresado");
    } else {
        boolean registrado = false;
        lblError.setText("");
        for (int i = 0; i < usr.length; i++) {
            if (usr[i] != null) {
                if (usr[i].getNombre_usuario().equals(txtNombre2.getText()) || usr[i].getCorreo().equals(txtCorreo.getText())
                    JOptionPane.showMessageDialog(null, "Error, ya hay un usuario con los mismos datos");
                    registrado = true;
                    break;
                }
            }
        }
```

otra validación que se agrego es que el usuario por obligación tenga que agregar ciertos patrones en la contraseña

```
if (!isValidPassword(txtContraseña.getText())) {
    JOptionPane.showMessageDialog(null, "Error, la contraseña debe contener los datos que se le piden");
    registrado = true;
}
```

Y que el campo para confirmar contraseña sea igual al campo de ingreso de contraseña.

```
if (!txtContraseña.getText().equals(txtConfirmacionContra.getText())) {
    JOptionPane.showMessageDialog(null, "Error, las contraseñas no coinciden");
    registrado = true;
}
```

En caso de que todos los datos sean validos, se ingresan los datos al arreglo correspondiente

```
if (!registrado) {
    usuario sr = new usuario();

    sr.setCorreo(txtCorreo.getText());
    sr.setNombre_usuario(txtNombre2.getText());
    sr.setApellido(txtApellido.getText());
    sr.setContraseña(txtContraseña.getText());
    sr.setConfirmacionContra(txtConfirmacionContra.getText());
    sr.setDPI((int) Long.parseLong(txtDPI.getText()));
    sr.setRol(cbRol.getSelectedItem().toString());
    sr.setFechaNacimiento(jdFechaNacimiento.getDateFormatString());
    sr.setGenero(cbGenero.getSelectedItem().toString());
    sr.setNacionalidad(cbNacionalidad.getSelectedItem().toString());
    sr.setSobrenombre(txtSobrenombre.getText());
    sr.setTelefono(Integer.parseInt(txtTelefono.getText()));
    if(opcionKiosko == true){
        sr.setCodigoKiosko(cbKiosko.getSelectedItem().toString());
    }
```

```
    for (int i = 0; i < usr.length; i++) {
        if (usr[i] == null) {
            JOptionPane.showMessageDialog(null, "Usuario registrado con exito");
            lblError.setForeground(Color.blue);
            usr[i] = sr;
            break;
        }
    }
    is.actualizarUsuario(usr);
}
```

En caso de que el usuario sea tipo kiosko, se tiene que mostrar los kioskos disponibles que se ingresaron por el administrador luego se tiene que ingresar que kiosko se asocio al usuario

```
private void cbRolActionPerformed(java.awt.event.ActionEvent evt) {

    if (cbRol.getSelectedItem().equals("Kiosko")) {
        lblKiosko.setVisible(true);
        cbKiosko.setVisible(true);
        opcionKiosko = true;
    } else {
        lblKiosko.setVisible(false);
        cbKiosko.setVisible(false);
    }
    cbKiosko.removeAllItems();
    for (int i = 0; i < kiosko.length; i++) {
        if (kiosko[i] != null) {
            cbKiosko.addItem(kiosko[i].getNombre());
        }
    }
}
Kioskos kio = new Kioskos();
```

Módulos para el administrador

Se programo el ingreso de los kioskos

```
        if (!ingresado) {
            Kioskos kio = new Kioskos();

            kio.setCodigoKiosko(txtCodigo.getText());
            kio.setNombre(txtNombreKiosko.getText());
            kio.setCodigoRegion(cbCodigoRegion.getSelectedItem().toString());

            for (int i = 0; i < kiosko.length; i++) {
                if (kiosko[i] == null) {
                    JOptionPane.showMessageDialog(null, "Kiosko registrado con exito");
                    kiosko[i] = kio;
                    break;
                }
            }
            is.actualizarKiosko(kiosko);
        }
    }
}
```

Se quemo en el sistema los siguientes datos:

Administrar Regiones

Regiones disponibles	Precio Estandar	Precio Especial
(M) Metropolitana	Q35.00	Q25.00
(NT) Norte	Q68.50	Q45.55
(NO) Nororiente	Q58.68	Q35.48
(SO) Suroriente	Q38.68	Q32.48
(SOC) Suroccidente	Q34.00	Q29.00
(NOC) Noroccidente	Q44.50	Q40.00

Se programo el ingreso de los departamentos y municipios mediante 2 botones

```
if (!depaIngresado) {
    Admin_Departamentos dp = new Admin_Departamentos();

    dp.setCodigoDepartamento(txtCodigoDepartamento.getText());
    dp.setNombreDepartamento(txtNombreDepartamento.getText());
    dp.setCodigoRegion(cbCodigoRegionP3.getSelectedItem().toString());

    for (int i = 0; i < depa.length; i++) {
        if (depa[i] == null) {
            JOptionPane.showMessageDialog(null, "Departamento registrado con exito");
            depa[i] = dp;
            break;
        }
    }
    lblDatos.setVisible(true);
    lblcodigoDepa.setVisible(true);
    cbCodigoDepartamento.setVisible(true);
    lblCodigoMuni.setVisible(true);
    txtCodigoMunicipio.setVisible(true);
    lblNombreMuni.setVisible(true);
    txtNombreMunicipio.setVisible(true);
    btnMunicipio.setVisible(true);
}
```

```
        if (!muniIngresado) {
            Admin_Municipios mn = new Admin_Municipios();

            mn.setCodigoMunicipio(txtCodigoMunicipio.getText());
            mn.setCodigoDepartamento(cbCodigoDepartamento.getSelectedItem().toString());
            mn.setNombreMunicipio(txtNombreMunicipio.getText());

            for (int i = 0; i < muni.length; i++) {
                if (muni[i] == null) {
                    JOptionPane.showMessageDialog(null, "Municipio registrado con exito");
                    muni[i] = mn;
                    break;
                }
            }
            is.actualizarMunicipios(muni);
        }
    }
}
```

Módulos para el usuario

Primero se registran los datos de las tarjetas de debito/crédito

```
        if (!tarjetaIngresada) {
            Tarjetas_debito tarD = new Tarjetas_debito();

            String numeroSeleccionado = txtNumeroTarjeta.getText();
            String ultimosCuatroDigitos = numeroSeleccionado.substring(numeroSeleccionado.length() - 4);
            String primerosSieteDigitosOfuscados = "*******" + ultimosCuatroDigitos;

            tarD.setNombre(txtNombreTarjeta.getText());
            tarD.setNumero(primerosSieteDigitosOfuscados);
            tarD.setFechaVencimiento(txtFechaVencimiento.getText());

            for (int i = 0; i < tar.length; i++) {
                if (tar[i] == null) {
                    JOptionPane.showMessageDialog(null, "Tarjeta registrada con exito");
                    tar[i] = tarD;
                    break;
                }
            }

        }
```

El siguiente código muestra los números de las tarjetas ingresadas en un combobox

```
cbTarjetasDisponibles6.removeAllItems();
for (int i = 0; i < tar.length; i++) {
    if (tar[i] != null) {
        cbTarjetasDisponibles6.addItem(tar[i].getNumero());
    }
}
is.actualizarTarjetas(tar);
```

Tambien se hizo el registro de los datos de facturación

```
        if (!ingresadofactu) {
            Datos_facturacion factu = new Datos_facturacion();

            factu.setNombreCliente(txtNombreCliente.getText());
            factu.setDireccion(txtDireccionCliente.getText());
            factu.setNit(txtNit.getText());

            for (int i = 0; i < fac.length; i++) {
                if (fac[i] == null) {
                    JOptionPane.showMessageDialog(null, "Datos registrados con exito");
                    fac[i] = factu;
                    break;
                }
            }
            is.actualizarDatosFacturacion(fac);

            lblNombreCliente.setText(txtNombreCliente.getText());
            lblDireccionCliente.setText(txtDireccionCliente.getText());
            lblNit.setText(txtNit.getText());
        }
    }
}
```

El siguiente código muestra los departamentos y municipios ingresados por el administrador en los combobox

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    cbDepartamentosOrigen.removeAllItems();
    cbMunicipiosDestino.removeAllItems();
    cbDepartamentosDestino.removeAllItems();
    cbMunicipiosOrigen.removeAllItems();
    for (int i = 0; i < admin.length; i++) {
        if (admin[i] != null) {
            cbDepartamentosOrigen.addItem(admin[i].getNombreDepartamento());
        }
    }

    for (int i = 0; i < admin.length; i++) {
        if (admin[i] != null) {
            cbDepartamentosDestino.addItem(admin[i].getNombreDepartamento());
        }
    }

    for (int i = 0; i < muni.length; i++) {
        if (muni[i] != null) {
            cbMunicipiosDestino.addItem(muni[i].getNombreMunicipio());
        }
    }
```

Luego se valido cual de los 3 checkbox del tipo de paquetes esta seleccionado

```
private void chPequenoActionPerformed(java.awt.event.ActionEvent evt) {
    chPequeno.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chPequeno.isSelected()) {
                chMediano.setEnabled(false);
                chGrande.setEnabled(false);
                elejido = "pequeño";
            } else {
                chMediano.setEnabled(true);
                chGrande.setEnabled(true);
            }
        }
    });

}
```

```
private void chMedianoActionPerformed(java.awt.event.ActionEvent evt) {
    chMediano.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chMediano.isSelected()) {
                chPequeno.setEnabled(false);
                chGrande.setEnabled(false);
                elejido = "Mediano";
            } else {
                chPequeno.setEnabled(true);
                chGrande.setEnabled(true);
            }
        }
    });
}
```

```java
private void chGrandeActionPerformed(java.awt.event.ActionEvent evt) {
    chGrande.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chGrande.isSelected()) {
                chPequeno.setEnabled(false);
                chMediano.setEnabled(false);
                elejido = "Grande";
            } else {
                chPequeno.setEnabled(true);
                chMediano.setEnabled(true);
            }
        }
    });
}
```

luego para realizar la cotización se realizo el siguiente código:

```java
private void btnCotizarActionPerformed(java.awt.event.ActionEvent evt) {

    if (txtNumeroPaquetes.getText().equals("")) {
        JOptionPane.showMessageDialog(null, "Hay algun dato que no ha sido ingresado");
    } else {
        int numPaquetes = Integer.parseInt(txtNumeroPaquetes.getText());
        String region = "";
        String texto = "";
        if (chPequeno.isSelected()) {
            for (int i = 0; i < admin.length; i++) {
                texto = admin[i].getNombreDepartamento();
                if (cbDepartamentosDestino.getSelectedItem().equals(texto)) {
                    region = admin[i].getCodigoRegion();
                    switch (region) {
                        case "M":
                            rEstandar = 35;
                            rEspecial = 25;
                            break;
```

```
        } else if (chGrande.isSelected()) {
            for (int i = 0; i < admin.length; i++) {
                texto = admin[i].getNombreDepartamento();
                if (cbDepartamentosDestino.getSelectedItem().equa
                    region = admin[i].getCodigoRegion();
                    switch (region) {
                        case "M":
                            rEstandar = 35;
                            rEspecial = 25;
                            break;
                        case "NT":
                            rEstandar = 68.50;
                            rEspecial = 45.55;
                            break;
```

```
                    case "NT":
                        rEstandar = 68.50;
                        rEspecial = 45.55;
                        break;
                    case "NO":
                        rEstandar = 58.68;
                        rEspecial = 35.48;
                        break;
                    case "SO":
                        rEstandar = 38.68;
                        rEspecial = 32.48;
                        break;
                    case "SOC":
                        rEstandar = 34;
                        rEspecial = 29;
                        break;
                    case "NOC":
                        rEstandar = 44.50;
                        rEspecial = 40;
                        break;
            }
```

```
                    case "NO":
                        rEstandar = 58.68;
                        rEspecial = 35.48;
                        break;
                    case "SO":
                        rEstandar = 38.68;
                        rEspecial = 32.48;
                        break;
                    case "SOC":
                        rEstandar = 34;
                        rEspecial = 29;
                        break;
                    case "NOC    ":
                        rEstandar = 44.50;
                        rEspecial = 40;
                        break;
                }
                totalEstandar = 15 * numPaquetes * rEstandar;
                totalEspecial = 15 * numPaquetes * rEspecial;
                break;
            }
        }
    }
```

Luego se valido que tipo de checkbox de servicio especial esta seleccionado y se mostraron los datos cotizados en los label correspondientes

```
private void chEstandarActionPerformed(java.awt.event.ActionEvent evt) {
    chEstandar.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chEstandar.isSelected()) {
                chEspecial.setEnabled(false);
                lblServicioEstandar1.setForeground(Color.blue);
            } else {
                chEspecial.setEnabled(true);
                lblServicioEstandar1.setForeground(Color.black);
            }
        }
    });
}
```

```java
private void chEspecialActionPerformed(java.awt.event.ActionEvent evt) {
    chEspecial.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (chEspecial.isSelected()) {
                chEstandar.setEnabled(false);
                lblServicioEspecial.setForeground(Color.blue);
            } else {
                chEstandar.setEnabled(true);
                lblServicioEspecial.setForeground(Color.black);
            }
        }
    });
}
```

```java
lblServicioEstandarl.setText(Double.toString(totalEstandar));
lblServicioEspecial.setText(Double.toString(totalEspecial));
```

Se valido nuevamente que tipo de checkbox del tipo de pago se selecciono

```java
private void checkCobraTarjetaActionPerformed(java.awt.event.ActionEvent evt) {
    checkCobraTarjeta.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (checkCobraTarjeta.isSelected()) {
                checkPagoEntrega.setEnabled(false);
            } else {
                checkPagoEntrega.setEnabled(true);
            }
            lblSeleccione.setVisible(true);
            cbTarjetasDisponibles6.setVisible(true);
            lblIngreseCodigo.setVisible(true);
            txtCVV.setVisible(true);
        }
    });
```

```
private void checkCobraTarjetaActionPerformed(java.awt.event.ActionEvent evt) {
    checkCobraTarjeta.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (checkCobraTarjeta.isSelected()) {
                checkPagoEntrega.setEnabled(false);
            } else {
                checkPagoEntrega.setEnabled(true);
            }
            lblSeleccione.setVisible(true);
            cbTarjetasDisponibles6.setVisible(true);
            lblIngreseCodigo.setVisible(true);
            txtCVV.setVisible(true);
        }
    });
}
```

Luego para realizar el pago, primero se valido que el numero de cvv estuviera lleno

```
if (checkPagoEntrega.isSelected()) {
    double Total = 0;
    if (chEstandar.isSelected()) {
        lblTipoServicio.setText("Estandar");
        Total = totalEstandar + 5;
        lblTotal.setText(Double.toString(Total));
    } else if (chEspecial.isSelected()) {
        lblTipoServicio.setText("Especial");
        Total = totalEspecial + 5;
        lblTotal.setText(Double.toString(Total));
    }
    tipo = "Pago contra entrega";
```

```
        } else {
            double Total = 0;
            if (chEstandar.isSelected()) {
                lblTipoServicio.setText("Estandar");
                Total = totalEstandar;
                lblTotal.setText(Double.toString(Total));
            } else if (chEspecial.isSelected()) {
                lblTipoServicio.setText("Especial");
                Total = totalEspecial;
                lblTotal.setText(Double.toString(Total));
            }
        }
        tipo = "Cobrar con tarjeta";
    }

    lbltiposervicioo.setText(lblTipoServicio.getText());
    lblDestino.setText(cbDepartamentosDestino.getSelectedItem().toString());
    lbltotaaal.setText(lblTotal.getText());
    lbltipooooo.setText(tipo);
}
```

```
lbltiposervicioo.setText(lblTipoServicio.getText());
lblDestino.setText(cbDepartamentosDestino.getSelectedItem().toString());
lbltotaaal.setText(lblTotal.getText());
lbltipooooo.setText(tipo);
```

Para la parte de descargar la guía, se genero un archivo html para mostrar los datos correspondientes

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String tablaP = "";
    // Generacion de html para reportes

    tablaP = "<table>"
            + "    <tr><th >" + "Codigo de paquete: No: 000123A2023" + "</th></tr>"
            + "    <tr><th>" + "Origen: " + cbDepartamentosOrigen.getSelectedItem().toString() + "/" + cbMunicipiosOrigen.get
            + "    <tr><th>" + "Destino: " + cbDepartamentosDestino.getSelectedItem().toString() + "/" + cbMunicipiosDestino.
            + "   <tr><th>" + "Tipo de pago: " + tipo + "</th></tr>"
            + "    <tr><th>" + "Tamaño del paquete: " + elejido + "</th></tr>"
            + "    <tr><th>" + "Numero de paquetes: " + txtNumeroPaquetes.getText() + "</th></tr>"
            + "    <tr><th>" + "Codigo de paquete: " + "IPClGlha7e" + "</th></tr>"
            + "    <tr><th>" + "Fecha de envio: " + "15/03/2023" + "</th></tr>"
            + "    <tr><th>" + "Total a pagar: " + lblTotal.getText() + "</th></tr>"
            + "  </table>";

    tablaP += "</table>";
    String estructura_html = "<html><body><h1>Factura</h1>" + tablaP + "</body></html>";
    File archivo = new File("reporte.html");
```

```
+ "/" + cbMunicipiosOrigen.getSelectedItem().toString() + "</th></tr>"
) + "/" + cbMunicipiosDestino.getSelectedItem().toString() + "</th></tr>"
```

```
    tablaP += "</table>";
    String estructura_html = "<html><body><h1>Factura</h1>" + tablaP + "</body></html>";
    File archivo = new File("reporte.html");
    try {

        //codigo para escribir el archivo
        archivo.createNewFile();
        FileWriter escritor_archivo = new FileWriter("reporte.html");
        escritor_archivo.write(estructura_html);
        escritor_archivo.close();
        //codigo para abrir el archivo creado
        Desktop abridor = Desktop.getDesktop();
        abridor.open(archivo);
    } catch (IOException ex) {
        java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
```

Para generar la factura en el html se hizo el mismo procedimiento

```
String tablaPrimera = "";
// Generacion de html para reportes

tablaPrimera = "<table>"
        + "    <tr><th >" + "NumFactura: No: 000123A2023" + "</th></tr>"
        + "    <tr><th>" + "Codigo paquete: IPClGlha7e" + "</th></tr>"
        + "    <tr><th>" + "Origen: " + cbDepartamentosOrigen.getSelectedItem().toString() + "</th></tr>"
        + "    <tr><th>" + "Destino: " + cbDepartamentosDestino.getSelectedItem().toString() + "</th></tr>"
        + "   <tr><th>" + "NIT: " + txtNit.getText() + "</th></tr>"
        + "    <tr><th>" + "Tipo de pago: " + tipo + "</th></tr>"
        + "  </table>";

String estructura_de_tabla_de_html = "<table><tr>\n"
        + "    <th style=\"border: 1px solid black; padding: 5px;\">Numero de paquete</th>\n"
        + "    <th style=\"border: 1px solid black; padding: 5px;\">Tamaño de paquete</th>\n"
        + "    <th style=\"border: 1px solid black; padding: 5px;\">Total de pago</th>\n"
        + "  </tr>";
```

```
        estructura_de_tabla_de_html += "<tr>"
                + "    <th style=\"border: 1px solid black; padding: 5px;\">" + txtNumeroPaquetes.getText() + "</th>"
                + "    <th style=\"border: 1px solid black; padding: 5px;\">" + elejido + "</th>"
                + "    <th style=\"border: 1px solid black; padding: 5px;\">" + lblTotal.getText() + "</th>"
                + "  </tr>";

        estructura_de_tabla_de_html += "</table>";
        String estructura_html = "<html><body><h1>Factura</h1>" + tablaPrimera + estructura_de_tabla_de_html + "</body></html>";
        File archivo = new File("reporte.html");
        try {

            //codigo para escribir el archivo
            archivo.createNewFile();
            FileWriter escritor_archivo = new FileWriter("reporte.html");
            escritor_archivo.write(estructura_html);
            escritor_archivo.close();
            //codigo para abrir el archivo creado
            Desktop abridor = Desktop.getDesktop();
            abridor.open(archivo);

        } catch (IOException ex) {
            java.util.logging.Logger.getLogger(Principal.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
```