

## Manual Técnico

Este manual este hecho con el propósito de que se puedan conocer los métodos y algoritmos realizados para crear el software Usac Delivery estos se hicieron con el lenguaje Java en el IDE NetBeans.

Este consiste principalmente de 3 paquetes Interfaz, Interfaz.usuario, imagen, ipc1.proyecto1.

**ipc1.proyecto1:** En este paquete se encuentran la clase main así como la creación de objetos junto con sus atributos y sus métodos.

- Clase IPC1PROYECTO1: dentro de estas se definen los ArrayList a utilizar dentro de el software así como la clase main donde se ejecutara la primera ventana que es Login para ingresar con el usuario.

```
package ipc1.proyecto1;
import Interfaz.Login;
import java.util.ArrayList;

public class IPC1PROYECTO1 {

    public static ArrayList<datoskiosco> kiosco = new ArrayList<>();
    public static ArrayList<Municipio> municipios = new ArrayList<>();
    public static ArrayList<usuariowithkiosco> uskiosco = new ArrayList<>();
    public static ArrayList<regionesyprecios> regyprec = new ArrayList<>(1);
    public static ArrayList<tarjeta> creditodebito = new ArrayList<>();
    public static ArrayList<Datosfactura> factura = new ArrayList<>();
    public static ArrayList<guiayfactura> facyguia = new ArrayList<>();

    public static void main(String[] args) {

        Login pantallalog = new Login();
        pantallalog.setVisible(true);
        pantallalog.setLocationRelativeTo(null);
    }
}
```

**Interfaz:** Dentro de este paquete se encuentran los frames pertenecientes al inicio de sesión y a la ventana de administrador.

- Clase Login: dentro de esta se encuentra un ArrayList de tipo usuario así como un usuario por defecto, en estas se encuentran los métodos para autenticación de usuarios también para guardar el usuario que se ingresa para esto para usarlo y mostrar datos para el usuario (usuario regular)

```

private void IngresarActionPerformed(java.awt.event.ActionEvent evt) {
    String usr = txtuser.getText();
    String contra = String.valueOf(txtpass.getPassword());
    esteusuario = txtuser.getText();
    if(usr.isEmpty() || contra.isEmpty()){
        JOptionPane.showMessageDialog(this,"Llene los campos vacios");
        return;
    }

    Usuario usuario = null;
    for(Usuario u : usuarios){
        if(u.getUsr().equals(usr) && u.getPass().equals(contra)){
            usuario = u;
            break;
        }
    }
    if(usuario == null){
        JOptionPane.showMessageDialog(this,"Usuario o contraseña incorrectos","Error", JOptionPane.ERROR_MESSAGE);
        txtuser.setText("");
        txtpass.setText("");
        return;
    }

    if(usuario.esAdmin){
        JOptionPane.showMessageDialog(this,"Bienvenido admin");
        AdminDisplay acceso = new AdminDisplay();
        acceso.setVisible(true);
        this.setVisible(false);
    }else{
        JOptionPane.showMessageDialog(this,"Bienvenido");

        Usuarioregular acceso = new Usuarioregular();
        acceso.setVisible(true);
        this.setVisible(false);
    }
}

private void vercontraActionPerformed(java.awt.event.ActionEvent evt) {
    if(vercontra.isSelected()){
        txtpass.setEchoChar((char)0);
    }else{
        txtpass.setEchoChar('\n2022');
    }
}

private void registraActionPerformed(java.awt.event.ActionEvent evt) {
    Registrouusuario acceso = new Registrouusuario();
    acceso.setVisible(true);
    this.setVisible(false);
}

public static String guardaresteusuario(){
    return esteusuario;
}

private boolean usuarioExiste(String usr) {
    for (Usuario u : usuarios) {
        if (u.getUsr().equals(usr)) {
            return true;
        }
    }
    return false;
}
}

```

- Clase Deptosymunicipios, este frame se gestionan los Departamentos y municipios a usar, Se llenan 2 combobox uno para la región y otro para eliminar usuarios así como se generan 2 números con la librería Random para asignarlo a el departamento y municipio, al llenar estos la tabla se llenara de los datos registrados.

```

package Interfaz;

import ipcl.proyecto1.IPC1PROYECTO1;
import static ipcl.proyecto1.IPC1PROYECTO1.municipios;
import ipcl.proyecto1.Municipio;
import java.util.Random;
import javax.swing.JOptionPane;
import javax.swing.table.TableModel;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author wilar
 */
public class Deptosymunicipios extends javax.swing.JFrame {

    /**
     * Creates new form Deptosymunicipios
     */
    public Deptosymunicipios() {
        initComponents();
        this.setLocationRelativeTo(null);
        llenarmunicipios();
        llenareliminar();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
        AdminDisplay acceso = new AdminDisplay();
        acceso.setVisible(true);
        this.setVisible(false);
    }

    private void ingresaActionPerformed(java.awt.event.ActionEvent evt) {
        //Departamento
        String namedepto = nombredepto.getText();
        String item = (String) regio.getSelectedItemAt();

        //Municipios
        String nombremun = nombremuni.getText();
        if (namedepto.isEmpty() || item.isEmpty() || nombremun.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Llene los espacios en blanco");
        }
        else if (namedepto.equals(nombremun)) {
            JOptionPane.showMessageDialog(this, "El nombre de municipio y departamento no pueden ser los mismos");
            nombredepto.setText("");
        }
        else {
            for (Municipio municipio : municipios) {
                if (municipio.getNombreMunicipio().equals(nombremun)) {
                    JOptionPane.showMessageDialog(this, "Este municipio ya está registrado");
                    nombremuni.setText("");
                    return;
                }
            }
            Random r = new Random();
            int coder = r.nextInt(9000)+1000;
            int deptoc = r.nextInt(900)+100;
            String codereg = String.valueOf(coder);
            String deptocode = String.valueOf(deptoc);
            Municipio nuevomun = new Municipio(codereg, item, namedepto, deptocode, nombremun);
            municipios.add(nuevomun);
            JOptionPane.showMessageDialog(null, "Datos registrados.");
            llenarmunicipios();
            llenareliminar();
            nombredepto.setText("");
            nombremuni.setText("");
        }
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
    }

    private void selecActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void eliminaregActionPerformed(java.awt.event.ActionEvent evt) {
        String it = (String) selec.getSelectedItemAt();
        for (int i = 0; i < municipios.size(); i++) {
            Municipio z = municipios.get(i);
            if (it.equals(z.getNombre() + " - " + z.getNombreMunicipio())) {
                JOptionPane.showMessageDialog(null, "Region eliminada");
                municipios.remove(i);
                llenarmunicipios();
                llenareliminar();
                break;
            }
        }
    }
}

```

```
private void llenarMunicipios() {
    DefaultTableModel modeloDefault = new DefaultTableModel(new String [] {"Codigo region", "Region", "Nombre Departamento", "Codigo Departamento", "Nombre municipio", "IPC1PROYECTO1.municipios.size() " });
    tablamanu.setModel(modeloDefault);

    TableModel modeloDatos = tablamanu.getModel();
    for(int i =0; i<IPC1PROYECTO1.municipios.size(); i++){
        Municipio nuevomun = IPC1PROYECTO1.municipios.get(i);
        modeloDatos.setValueAt(nuevomun.getCodigoRegion(), i, 0);
        modeloDatos.setValueAt(nuevomun.getRegion(), i, 1);
        modeloDatos.setValueAt(nuevomun.getNombre(), i, 2);
        modeloDatos.setValueAt(nuevomun.getCodigoDepartamento(), i, 3);
        modeloDatos.setValueAt(nuevomun.getNombreDepartamento(), i, 4);
    }
}

public void llenarEliminar() {
    selec.removeAllItems();
    if(municipios.isEmpty()) {
        selec.addItem("No hay datos");
    } else {
        for(Municipio z: municipios) {
            selec.addItem(z.getNombre()+" - "+z.getNombreMunicipio());
        }
    }
}
```

- Clase manejokiosko: En esta clase se agregara un kiosco a una región mediante el combobox para llenar esta, la cual extraerá el código de región anteriormente creado en la clase Deptosymunicipios, por lo que si no se llenan estos antes no podrá agregar nuevos kioscos, asimismo un comobox que llena los datos de los kioscos registrados para que puedan ser eliminados.

```

package Interfaz;

import icpl.proyecto1.IPC1PROYECT01;
import static icpl.proyecto1.IPC1PROYECT01.kiosco;
import static icpl.proyecto1.IPC1PROYECT01.municipios;
import icpl.proyecto1.Municipio;
import icpl.proyecto1.datoskiosco;
import java.util.Random;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

/**
 *
 * @author wilar
 */
public class manejojokiosko extends javax.swing.JFrame {

    public manejojokiosko() {
        initComponents();
        this.setLocationRelativeTo(null);
        llenarkiosco();
        llenareliminar();
    }

    @SuppressWarnings("unchecked")
    // 
    Generated Code

    private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
        AdminDisplay next = new AdminDisplay();
        next.setVisible(true);
        this.setVisible(false);
    }

    private void regiozzActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        llenarregio();
    }

    private void asignacionActionPerformed(java.awt.event.ActionEvent evt) {
        //Assign variables

        String namekios = kiosco.getText();
        String item = (String) regiozz.getSelectedItemAt();

        if (item.isEmpty() || namekios.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Llene los espacios en blanco");
        }
    }

```

```

    } else {
        for (datoskiosco k : kiosco) {
            if (k.getNombrekiosco().equals(namekios)) {
                JOptionPane.showMessageDialog(this, "Este municipio ya está registrado");
                kiosname.setText("");
                return;
            }
        }

        String codigoregional = obtenerCodigoRegional(item);
        Random r = new Random();
        int numrand = r.nextInt(90000)+10000;
        String codigokios = String.valueOf(numrand);
        datoskiosco nuevokiosco = new datoskiosco(codigokios, namekios, codigoregional);
        kiosco.add(nuevokiosco);
        JOptionPane.showMessageDialog(null, "Kiosco Registrado.");
        kiosname.setText("");
        llenarkiosco();
        llenareliminar();
    }
}

private void kiosnameActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void eliminarActionPerformed(java.awt.event.ActionEvent evt) {
    String it = (String) kioscoborrar.getSelectedItemAt();
    for (int i = 0; i < kiosco.size(); i++) {
        datoskiosco z = kiosco.get(i);
        if (it.equals(z.getCodigoKiosco()+" - "+z.getNombrekiosco())) {
            JOptionPane.showMessageDialog(null, "Kiosco eliminado");
            kiosco.remove(i);
            llenarkiosco();
            llenareliminar();
            break;
        }
    }
}

public void llenarregio() {
    regiozz.removeAllItems();
    if (municipio.isEmpty()) {
        regiozz.addItem("No hay datos");
    } else {
        for (Municipio mun : municipios) {
            String codigoregion = mun.getCodigoReg();
            String region = mun.getRegion();
            String item = codigoregion + " - " + region;
            regiozz.addItem(item);
        }
    }
}

```

```

private String obtenerCodigoRegional(String item) {
    String[] partes = item.split("- ");
    if (partes.length != 2) {
        throw new IllegalArgumentException("Este elemento no tiene el formato esperado");
    }
    return partes[0];
}

private void llenarkiosco() {
    DefaultTableModel modeloDefault = new DefaultTableModel(new String [] {"Codigo de kiosco", "Nombre del kiosco", "Codigo de region", IPCI.PROYECTO1.kiosco.size() });
    tablaKiosco.setModel(modeloDefault);

    TableModel modeloDatos = tablaKiosco.getModel();
    for(int i =0; i<IPCI.PROYECTO1.kiosco.size(); i++){
        datosKiosco nuevoKiosco = IPCI.PROYECTO1.kiosco.get(i);
        modeloDatos.setValueAt(nuevoKiosco.getCodigoKiosco(), i, 0);
        modeloDatos.setValueAt(nuevoKiosco.getNombreKiosco(), i, 1);
        modeloDatos.setValueAt(nuevoKiosco.getCodigoRegionKiosco(), i, 2);
    }
}

public void llenareliminar() {
    kioscoBorrar.removeAllItems();
    if(kiosco.isEmpty()) {
        kioscoBorrar.addItem("No hay datos");
    } else {
        for(datosKiosco d : kiosco) {
            kioscoBorrar.addItem(d.getCodigoKiosco()+" - "+d.getNombreKiosco());
        }
    }
}

```

- Clase manejuregionyprecio: en esta clase se definen las regiones y los precios por defecto por lo que esta se deberá de abrir para registrar los datos predeterminados, esta contiene un botón limpiar para que cuando el usuario lo presione los datos se borren y el usuario pueda agregar nuevos precios.

```

package Interfaz;

import static ipci.proyecto1.IPCI.PROYECTO1.regiones;
import ipci.proyecto1.regionesyprecios;
import java.swing.JOptionPane;

/**
 *
 * @author wilmar
 */
public class manejuregionyprecio extends javax.swing.JFrame {

    public manejuregionyprecio() {
        initComponents();
        this.setLocationRelativeTo(null);
        regiones.add(new regionesyprecios("00 Metropolitana",35.00, 25.00, "00T Norte", 45.00, 45.55, "00O Occidente", 55.45, 35.45, "00S Suriente", 30.45, 32.45, "00C Sucooccidente", 34.00, 29.00, "00C1"));
    }

    @SuppressWarnings("unchecked")
    // Generated Code

    private void datosregionActionPerformed(java.awt.event.ActionEvent evt) {
        //Region metropolitana
        String metropolitana = "(0) Metropolitana";
        String metestandar = metroestandar.getText();
        String metespecial = metroespecial.getText();
        //Region norte
        String nt = "00T Norte";
        String ntestandar = norteestandar.getText();
        String ntespecial = norteespecial.getText();
        //Region occidente
        String oo = "00O Occidente";
        String oostandar = oostestandar.getText();
        String ooespecial = oostespecial.getText();
        //Region suriente
        String so = "00S Suriente";
        String soestandar = soestestandar.getText();
        String soespecial = soespecial.getText();
        //Region sucooccidente
        String soo = "00C Sucooccidente";
        String socesta = socestestandar.getText();
        String soceesp = soceespecial.getText();
        //Region sucooccidente
        String sco = "00C1 Sucooccidente";
        String socesta = socestestandar.getText();
        String socesp = socespecial.getText();

        if (metestandar.isEmpty() || metespecial.isEmpty() || ntestandar.isEmpty() || ntespecial.isEmpty() || oostandar.isEmpty() || oostespecial.isEmpty() || soestandar.isEmpty() || soespecial.isEmpty() || socesta.isEmpty() || socesta.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Debesa llenar los espacios en blanco", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (regiones.size() >= 1) {
            JOptionPane.showMessageDialog(null, "Las regiones ya estan registradas", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            regionesyprecios nuevoregion = new regionesyprecios(metropolitana, Double.parseDouble(metestandar), Double.parseDouble(metespecial), nt, Double.parseDouble(ntestandar), Double.parseDouble(ntespecial), no, Double.parseDouble(nostestandar), Double.parseDouble(notespecial));
            regiones.add(nuevoregion);
            JOptionPane.showMessageDialog(null, "Datos registrados");
        }
    }

    private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
        AdminDisplay acceso = new AdminDisplay();
        acceso.setVisible(true);
        this.setVisible(false);
    }

    private void limpiarActionPerformed(java.awt.event.ActionEvent evt) {
        if (regiones.isEmpty()) {
            JOptionPane.showMessageDialog(null, "El registro de regiones y precios ya está vacío");
        } else {
            metroestandar.setText("");
            metroespecial.setText("");
            norteestandar.setText("");
            norteespecial.setText("");
            oostestandar.setText("");
            oostespecial.setText("");
            soestandar.setText("");
            soespecial.setText("");
            socestestandar.setText("");
            socestespecial.setText("");
            socestestandar.setText("");
            socestespecial.setText("");
            regiones.clear();
        }
    }
}

```

- Clase Reportes: Dentro de esta clase se llenara la tabla para organizar los datos de facturación de mayor a menor.

```
package Interfaz;

import static ipc1.proyecto1.IPC1PROYECTO1.facyguia;
import ipc1.proyecto1.guiayfactura;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author wilar
 */

public class Reportes extends javax.swing.JFrame {

    public Reportes() {
        initComponents();
        this.setLocationRelativeTo(null);

        DefaultTableModel modelo = (DefaultTableModel) reportes.getModel();

        List<Object[]> filas = new ArrayList<>();
        for (guiayfactura factura : facyguia) {
            String totalString = String.format("%.2f", factura.getTotal());
            Object[] fila = new Object[4];
            fila[0] = factura.getDestino();
            fila[1] = factura.getNpaquete();
            fila[2] = totalString;
            fila[3] = factura.getUsuariofactura();
            filas.add(fila);
        }

        Comparator<Object[]> comparador = new Comparator<Object[]>() {
            @Override
            public int compare(Object[] fila1, Object[] fila2) {
                double total1 = Double.parseDouble((String) fila1[2]);
                double total2 = Double.parseDouble((String) fila2[2]);
                return Double.compare(total2, total1);
            }
        };

        Collections.sort(filas, comparador);

        for (Object[] fila : filas) {
            modelo.addRow(fila);
        }

        reportes.setModel(modelo);
    }
}
```

**Paquete Interfaz.usuario:** dentro de este paquete se encuentran las clases para el registro de datos de facturación tarjeta de crédito y debito, cotización.

- Clase Registrodetarjetas: dentro de esta clase se registraran las tarjetas que el usuario usara para la realizar una venta.

```

private void registrarActionPerformed(java.awt.event.ActionEvent evt) {
    String ntar = nombretarjeta.getText();
    String numtar = notarjeta.getText();
    String venc = fechavencimiento.getText();
    Pattern patronvencimiento = Pattern.compile("^\\d{2}/\\d{2}$");

    if(!patronvencimiento.matcher(venc).matches()) {
        JOptionPane.showMessageDialog(null, "El formato de la fecha es incorrecto. Utilice el formato dd/mm/aaaa");
        fechavencimiento.setText("");
        return;
    }

    if (ntar.isEmpty() || numtar.isEmpty() || venc.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Necesita llenar los espacios en blanco", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(numtar.length()<16 || numtar.length()>16 ){
        JOptionPane.showMessageDialog(null, "El numero de la tarjeta debe de ser de 16 digitos.");
        notarjeta.setText("");
    }else{
        for(int i = 0; i<creditodebito.size(); i++){
            tarjeta tar = creditodebito.get(i);

            if(tar.getNumerotarjeta().equals(numtar)){
                JOptionPane.showMessageDialog(this, "Esta tarjeta ya está registrada");

                break;
            }
        }

        tarjeta nuevatar = new tarjeta(ntar, numtar, venc, ntar);
        creditodebito.add(nuevatar);
        JOptionPane.showMessageDialog(null, "Tarjeta registrada");
        llenareliminar();
        nombretarjeta.setText("");
        notarjeta.setText("");
        fechavencimiento.setText("");
    }
}

```

```

private void eltarActionPerformed(java.awt.event.ActionEvent evt) {
    String it = (String) eliminartarjeta.getSelectedItemAt();
    for (int i = 0; i < creditodebito.size(); i++) {
        tarjeta z = creditodebito.get(i);
        if (it.equals(z.getNumerotarjetatruncado())) {
            JOptionPane.showMessageDialog(null, "Tarjeta Eliminada");
            creditodebito.remove(i);
            llenareliminar();
            break;
        }
    }
}

```

```

public void llenareliminar(){

    eliminartarjeta.removeAllItems();
    if(creditodebito.isEmpty()) {
        eliminartarjeta.addItem("No hay datos");
    } else {
        for(tarjeta z: creditodebito) {
            eliminartarjeta.addItem(z.getNumerotarjetatruncado());
        }
    }
}

```

- Clase datosdefacturacion: en esta clase hay métodos para almacenar los datos de facturación que el usuario ingrese para sus clientes, este tendrá un combo box y un botón para eliminar estos datos.

```
private void guardarActionPerformed(java.awt.event.ActionEvent evt) {
    String name = nombre.getText();
    String direcc = direccion.getText();
    String nt = nits.getText();

    for (Datosfactura f:factura ) {
        if (nombre.equals(f.getNombrecliente()) && direcc.equals(f.getDireccion()) && nt.equals(f.getNit())) {
            JOptionPane.showMessageDialog(this, "Este cliente ya ha sido registrado");
            return;
        }
    }

    if (name.isEmpty() || direcc.isEmpty() || nt.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Debe de llenar todos los campos vacios");
    }

    else if (nt.length() < 9 || nt.length() > 9) {
        JOptionPane.showMessageDialog(this, "El numero de nit debe de ser de 9 digitos");
    }
    else {
        Datosfactura datos = new Datosfactura(name, direcc, nt);
        factura.add(datos);
        JOptionPane.showMessageDialog(this, "Cliente registrado");
        llenareliminar();
        nombre.setText("");
        direccion.setText("");
        nits.setText("");
    }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Usuarioregular acceso = new Usuarioregular();
    acceso.setVisible(true);
    this.setVisible(false);
}

private void elimActionPerformed(java.awt.event.ActionEvent evt) {
    String it = (String) elimfact.getSelectedItem();
    for (int i = 0; i < factura.size(); i++) {
        Datosfactura z = factura.get(i);
        if (it.equals(z.getNombrecliente() + " - " + z.getDireccion())) {
            JOptionPane.showMessageDialog(null, "Datos eliminados");
            factura.remove(i);
            llenareliminar();
            break;
        }
    }
}

public void llenareliminar() {
    elimfact.removeAllItems();
    if (factura.isEmpty()) {
        elimfact.addItem("No hay datos");
    }
    else {
        for (Datosfactura z: factura) {
            elimfact.addItem(z.getNombrecliente() + " - " + z.getDireccion());
        }
    }
}
```

- Clase cotización: Dentro de esta clase se encuentra la cotización y facturación en donde se usaran los datos almacenados de el paquete Interfaz y de los datos añadidos anteriormente en este mismo paquete por lo que si no se ha ingresado datos no podrá realizar cotizaciones, este consiste en 4 combo box para la selección de municipios y departamentos de origen y destino, un llenado de número de paquetes y tamaño y el botón cotiza que calcula el precio, luego procederá a seleccionar el tipo de pago para así poder ver la factura o la guía, esta se puede cancelar si lo desea aunque si visualiza la factura y regresa y le da a cancelar ya no la podrá cancelar.



```

package Interfaz.usuario;

import ipcl.proyectol.Datosfactura;
import static ipcl.proyectol.IPC1PROYECTOL.creditodebito;
import static ipcl.proyectol.IPC1PROYECTOL.factura;
import static ipcl.proyectol.IPC1PROYECTOL.municipios;
import static ipcl.proyectol.IPC1PROYECTOL.regyprec;
import ipcl.proyectol.Municipio;
import Interfaz.Login;
import static ipcl.proyectol.IPC1PROYECTOL.facyguia;

import static ipcl.proyectol.IPC1PROYECTOL.uskiosco;
import ipcl.proyectol.guiayfactura;
import ipcl.proyectol.regionesyprecios;
import ipcl.proyectol.tarjeta;
import ipcl.proyectol.usuariowithkiosco;
import java.awt.event.ActionEvent;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Random;

import javax.swing.JOptionPane;

/**
 *
 * @author wilar
 */
public class cotizacion extends javax.swing.JFrame {

    public cotizacion() {
        initComponents();
        this.setLocationRelativeTo(null);
        llenarcab();
        llenarar();
        llenarestandar();
        llenarespecial();
        llenarorigen();
        llenardestino();
        //Botones de cobros y servicios
        cobros.add(contraentrega);
        cobros.add(pagotarjeta);

        servicios.add(standar);
        servicios.add(special);
    }

```

```

        servicios.add(standar);
        servicios.add(special);

        pagotarjeta.addActionListener((ActionEvent e) -> {
            notatarjeta.setEnabled(true);
            pagotarjeta.setEnabled(false);
            cvv.setEnabled(true);
        });

        contraentrega.addActionListener((ActionEvent e) -> {
            notatarjeta.setEnabled(false);
            contraentrega.setEnabled(true);
            cvv.setEnabled(false);
            pagotarjeta.setEnabled(true);
        });

        special.addActionListener((ActionEvent e) -> {
            servestandar.setEnabled(false);
            servspecial.setEnabled(true);
        });

        standar.addActionListener((ActionEvent e) -> {
            servspecial.setEnabled(false);
            servestandar.setEnabled(true);
        });

        servspecial.setEnabled(false);
        servestandar.setEnabled(true);
    }

    double totalCalculado = 0;
    String tiposervicio="";
    String usuariodesto= "";

```

```

private void cotisaActionPerformed(java.awt.event.ActionEvent evt) {

    String paquet = paquetes.getText();
    String sispa = tamahopa.getText();
    String dest = (String) deptodestino.getSelectedItemAt();
    String deptorig = (String) deptorigen.getSelectedItemAt();
    String munor = (String) munorigen.getSelectedItemAt();
    String mundes = (String) mundeestino.getSelectedItemAt();
    String seres = (String) servestandar.getSelectedItemAt();
    String seresp = (String) servespecial.getSelectedItemAt();

    if (paquet.isEmpty() || sispa.isEmpty() || servicios.getSelection() == null) {
        JOptionPane.showMessageDialog(this, "Debe de llenar todos los campos vacios");
        return;
    }

    if (dest.equals("No hay datos") || deptorig.equals("No hay datos") || munor.equals("No hay datos") || mundes.equals("No hay datos")) {
        JOptionPane.showMessageDialog(this, "Debe de registrar Departamentos y Municipios");
        return;
    }

    if (seres.equals("No hay datos") || seresp.equals("No hay datos")) {
        JOptionPane.showMessageDialog(this, "Debe de registrar precios de regiones y departamentos");
        return;
    }

    if (munor.equals(mundes)) {
        JOptionPane.showMessageDialog(this, "El municipio no puede ser el mismo");
        return;
    }

    String cadenaEs = seres.split(" - ")[0];
    String cadenaEsp = seresp.split(" - ")[0];
    double precios = Double.parseDouble(cadenaEs);
    double preciosp = Double.parseDouble(cadenaEsp);
    double nopaket;
    double paquetesise;
    String[] seresarray = seres.split(" - ");
    String[] serespharray = seresp.split(" - ");
    String[] destarray = dest.split(" - ");

    if (sestandar.isSelected() && seresarray.length > 1 && !seresarray[1].equals(destarray[1])) {
        JOptionPane.showMessageDialog(this, "El destino seleccionado no coincide con el destino del servicio estándar");
        return;
    }

    if (sespecial.isSelected() && serespharray.length > 1 && !serespharray[1].equals(destarray[1])) {
        JOptionPane.showMessageDialog(this, "El destino seleccionado no coincide con el destino del servicio especial");
        return;
    }

    nopaket = Double.parseDouble(paquet);
    paquetesise = Double.parseDouble(sispa);

    if (sestandar.isSelected()) {
        tiposervicio = "Servicio estandar";
        totalCalculado = nopaket * paquetesise * precios;
    } else if (sespecial.isSelected()) {
        tiposervicio = "Servicio especial";
        totalCalculado = nopaket * paquetesise * preciosp;
    }

    String totalString = String.format("%.2f", totalCalculado);
    total.setText("Q. " + totalString);
}

```

```

private void realizarPagoActionPerformed(java.awt.event.ActionEvent evt) {
    String dest = (String) deptoDestino.getSelectedItemAt();
    String paquet = paquetes.getText();
    String deptorig = (String) deptorigen.getSelectedItemAt();
    String munor = (String) munorigen.getSelectedItemAt();
    String mundes = (String) mundeDestino.getSelectedItemAt();
    String sispa = tamanoPaq.getText();
    String ntar = (String) notarjeta.getSelectedItemAt();
    String css = cvv.getText();
    String datosfa = (String) facturacion.getSelectedItemAt();
    String usuarioactual = Login.guardarestusuuario();

    Random random = new Random();
    Calendar calendar = Calendar.getInstance();
    Date fechaActual = calendar.getTime();
    String datof = String.valueOf(facturacion.getSelectedItemAt());
    String[] datos = datof.split("-");
    String nombre = datos[0];
    String nit = datos[1];
    String direccion = datos[2];

    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    String fechaActualString = formatoFecha.format(fechaActual);

    String letras = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_";
    String codigopaqAleatorio = "IPC1";
    for (int i = 0; i < 5; i++) {
        int index = random.nextInt(letras.length());
        codigopaqAleatorio += letras.charAt(index);
    }

    for (Municipio m: municipios) {
        for (usuarioWithkiosco u: usuarios) {
            String iguales = m.getRegion();
            if (iguales.equals(u.getCoder())) {
                usuariodesto = u.getUser();
            }
        }
    }

    int nfacturaAleatorio = random.nextInt(90000) + 10000;

    if (cobros.getSelection() == null) {
        JOptionPane.showMessageDialog(this, "Seleccione un método de pago");
        return;
    }

    if (contraentrega.isSelected()) {
        totalCalculado += 5;
        JOptionPane.showMessageDialog(this, "El total a pagar es: Q. " + totalCalculado);

        guiyayfactura nuevo = new guiyayfactura(usuarioactual, usuariodesto, String.valueOf(nfacturaAleatorio), codigopaqAleatorio, deptorig, munor, dest, mundes, nit, non
        factyguia.add(nuevo);
    } else if (pagotarjeta.isSelected()) {

        if (ntar.isEmpty() || css.isEmpty() || datosfa.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Llene los campos vacios");
        } else {
            JOptionPane.showMessageDialog(this, "Se ha realizado el cobro de Q. " + totalCalculado + " a la tarjeta " + ntar);
            guiyayfactura nuevo = new guiyayfactura(usuarioactual, usuariodesto, String.valueOf(nfacturaAleatorio), codigopaqAleatorio, deptorig, munor, dest, mundes, nit, non
            factyguia.add(nuevo);
        }
    }
}

```

```

private void cancelaActionPerformed(java.awt.event.ActionEvent evt) {
    int ultimoIndex = factyguia.size() - 1;
    if (ultimoIndex < 0) {
        JOptionPane.showMessageDialog(this, "No hay envíos para cancelar");
        return;
    }
    factyguia.remove(ultimoIndex);
    JOptionPane.showMessageDialog(this, "Envío cancelado");
}

private void facturadescargaActionPerformed(java.awt.event.ActionEvent evt) {
    compras vi = new compras();
    vi.setVisible(true);
    this.setVisible(false);
}

private void guiadescargaActionPerformed(java.awt.event.ActionEvent evt) {
    descargaguia vi = new descargaguia(factyguia);
    vi.setVisible(true);
    this.setVisible(false);
}

//Llenar tarjeta
public void llenarcard(){

    for (tarjeta s : creditodebito) {
        String numeroCompleto = s.getNumerotarjeta();
        String ultimosCuatroDigitos = numeroCompleto.substring(numeroCompleto.length() - 4);
        String mascara = "";
        for (int i = 0; i < numeroCompleto.length() - 4; i++) {
            mascara += "x";
        }
        s.setNumerotarjetatruncado(mascara + ultimosCuatroDigitos);
    }
    notarjeta.removeAllItems();
    if(creditodebito.isEmpty()) {
        notarjeta.addItem("No hay datos");
    } else {
        for(tarjeta s: creditodebito) {
            notarjeta.addItem(s.getNumerotarjetatruncado());
        }
    }
}

//Llenar facturacion
public void llenarf(){

    facturacion.removeAllItems();
    if(factura.isEmpty()) {
        facturacion.addItem("No hay datos");
    } else {
        for(Datosfactura s: factura) {
            facturacion.addItem(s.getNombrecliente() + " - " + s.getNit() + " - " + s.getDireccion());
        }
    }
}
}

```

- Clase compras: esta clase contiene un frame para visualizar la factura anteriormente creada y esta contiene un botón para descargar el archivo en html.

```
private void mostrarFacturaActual() {
    if (facyguia.isEmpty() || indiceFacturaActual >= facyguia.size()) {
        JOptionPane.showMessageDialog(this, "No hay más facturas");
        return;
    }

    // Obtener la factura actual
    guiyfactura factura = facyguia.get(indiceFacturaActual);
    String totalString = String.format("%.2f", factura.getTotal());
    // Llenar los campos de texto con los datos de factura
    llenanit.setText(factura.getNitz());
    llenanombre.setText(factura.getCliente());
    llenadireccion.setText(factura.getDireccioncliente());
    llenapago.setText(factura.getTipopago());
    llenafact.setText(factura.getNfactura());

    DefaultTableModel modeloTabla = new DefaultTableModel();

    modeloTabla.addColumn("Número de Paquetes");
    modeloTabla.addColumn("Tamaño del Paquete");
    modeloTabla.addColumn("Total de Pago");

    Object[] fila = {
        factura.getNpaquete(),
        factura.getTamapaquete(),
        totalString
    };
    modeloTabla.addRow(fila);

    tabladatos.setModel(modeloTabla);
}
```

```
private void regresaActionPerformed(java.awt.event.ActionEvent evt) {
    cotizacion acceso = new cotizacion();
    acceso.setVisible(true);
    this.setVisible(false);
}
```

```
private void anteriorActionPerformed(java.awt.event.ActionEvent evt) {
    if (indiceFacturaActual > 0) {
        indiceFacturaActual--;
        mostrarFacturaActual();
    }
}
```

```
private void siguienteActionPerformed(java.awt.event.ActionEvent evt) {
    if (indiceFacturaActual + 1 < facyguia.size()) {
        indiceFacturaActual++;
        mostrarFacturaActual();
    }
}
```

```

private void descargarFactActionPerformed(java.awt.event.ActionEvent evt) {
    if (facyguia.isEmpty()) {
        JOptionPane.showMessageDialog(this, "No hay facturas");
        return;
    }

    guiyafactura factura = facyguia.get(indiceFacturaActual);

    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Guardar Factura");
    fileChooser.setSelectedFile(new File("factura.html"));

    int userSelection = fileChooser.showSaveDialog(this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        String filePath = fileToSave.getAbsolutePath();

        try {
            FileWriter fileWriter = new FileWriter(filePath);
            fileWriter.write("<html>\n<head>\n<title>Factura</title>\n</head>\n<body>\n");
            fileWriter.write("<h1>Factura</h1>\n");
            fileWriter.write("<p><strong>NIT:</strong> " + factura.getNitz() + "</p>\n");
            fileWriter.write("<p><strong>Nombre:</strong> " + factura.getClientes() + "</p>\n");
            fileWriter.write("<p><strong>Dirección:</strong> " + factura.getDireccioncliente() + "</p>\n");
            fileWriter.write("<p><strong>Tipo de pago:</strong> " + factura.getTipopago() + "</p>\n");
            fileWriter.write("<p><strong>Factura no:</strong> " + factura.getNfactura() + "</p>\n");
            fileWriter.write("<table>\n<thead>\n<tr><th>Número de Paquetes</th><th>Tamaño del Paquete</th><th>Total de");
            fileWriter.write("<tr><td>" + factura.getNpaquete() + "</td><td>" + factura.getTamapaquete() + "</td><td>");
            fileWriter.write("</tbody>\n</table>\n");
            fileWriter.write("</body>\n</html>");
            fileWriter.close();

            JOptionPane.showMessageDialog(this, "Factura guardada exitosamente");

        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Error al guardar la factura: " + ex.getMessage());
        }
    }
}

```

- Clase descargaguia: dentro de esta clase hay un frame que muestra información de la guía del paquete con su código de barras, este de la misma manera que en factura se podrá descargar en formato html para su visualización.

```

public class descargaguias extends javax.swing.JFrame {
    private int indiceActual = 0;
    private ArrayList<guiayfactura> facyguia;

    public descargaguias(ArrayList facyguia) {

        initComponents();
        this.setLocationRelativeTo(null);
        this.facyguia = facyguia;
        actualizarCampos();

        ImageIcon codigobara = new ImageIcon("src/imagen/codigobarras.jpg");
        Icon icono = new ImageIcon(codigobara.getImage().getScaledInstance(codigobar.getWidth(), codigobar.getHeight(), Image.SCALE_DEFAULT));

        codigobar.setIcon(icono);
        this.repaint();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void anteriorActionPerformed(java.awt.event.ActionEvent evt) {
        anterior.addActionListener((e) -> {
            indiceActual--;
            if (indiceActual < 0) {
                indiceActual = facyguia.size() - 1;
            }
            actualizarCampos();
        });
    }

    private void siguienteActionPerformed(java.awt.event.ActionEvent evt) {
        siguiente.addActionListener((e) -> {
            indiceActual++;
            if (indiceActual >= facyguia.size()) {
                indiceActual = 0;
            }
            actualizarCampos();
        });
    }

    private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
        cotizacion vi = new cotizacion();
        vi.setVisible(true);
        this.setVisible(false);
    }

    private void descargaActionPerformed(java.awt.event.ActionEvent evt) {
        if (facyguia.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No hay Guías");
            return;
        }
    }

```

```

    private void actualizarCampos() {

        if (facyguia.isEmpty() || indiceActual >= facyguia.size()) {
            JOptionPane.showMessageDialog(this, "No hay más Guías");
            return;
        }

        guiayfactura actual = facyguia.get(indiceActual);
        llenaaorigen.setText(actual.getOrigen());
        llenadestino.setText(actual.getDestino());
        codigopa.setText(actual.getCodigopa());
        nopaquetes.setText(actual.getNpaquete());
        tamañopaquetes.setText(actual.getTamapaquete() + " - " + "1b");
        remitente.setText(actual.getUsuariofactura());
        destinatarioz.setText(actual.getUsuariodest());
    }

```

```

private void descargarHTML() {
    ImageIcon codigobara = new ImageIcon(getClass().getResource("/imagen/codigobarras.jpg"));
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Guardar archivo");
    fileChooser.setSelectedFile(new File("guia.html"));
    int userSelection = fileChooser.showSaveDialog(this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        String filePath = fileToSave.getAbsolutePath();
        try {
            FileWriter fileWriter = new FileWriter(filePath);
            fileWriter.write("<html>");

            fileWriter.write("<head>");

            fileWriter.write("<title>Guia</title>");

            fileWriter.write("</head>");

            fileWriter.write("<body>");

            fileWriter.write("<h1>Informacion de la Guia</h1>");

            fileWriter.write("<p><strong>Origen:</strong> " + llenaaorigen.getText() + "</p>");

            fileWriter.write("<p><strong>Destino:</strong> " + llenadestino.getText() + "</p>");

            fileWriter.write("<p><strong>Remitente:</strong> " + remitente.getText() + "</p>");

            fileWriter.write("<p><strong>Destinatario:</strong> " + destinatarios.getText() + "</p>");

            fileWriter.write("<p><strong>Codigo del paquete:</strong> " + codigopa.getText() + "</p>");

            fileWriter.write("<p><strong>Numero de paquetes:</strong> " + nopaquetes.getText() + "</p>");

            fileWriter.write("<p><strong>Tamaño del paquete:</strong> " + tamañopaquetes.getText() + "</p>");

            fileWriter.write("</body>");

            fileWriter.write("<p><strong>Codigo de barras:</strong> <img src='" + codigobara.toString() + "'/></p>");

            fileWriter.write("</html>");

            fileWriter.close();

            JOptionPane.showMessageDialog(this, "Factura guardada exitosamente");

        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Error al guardar la factura: " + ex.getMessage());
        }
    }
}

```



- Clase verenvios: dentro de esta clase hay un frame con una tabla que muestra los datos de envío que se guardaran en el arraylist facyguia.

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

    public verenvios() {
        initComponents();
        this.setLocationRelativeTo(null);

        for (guiayfactura e: facyguia) {
            Object[] fila = new Object[]{
                e.getCodigopaq(),
                e.getServicetype(),
                e.getUsuariodest(),
                e.getTotal(),
                e.getTipopago()
            };
            model.addRow(fila);
        }

        envios.setModel(model);

    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void regrearActionPerformed(java.awt.event.ActionEvent evt) {
        Usuarioregular acceso = new Usuarioregular();
        acceso.setVisible(true);
        this.setVisible(false);
    }

    DefaultTableModel model = new DefaultTableModel(
        new Object[][] {},
        new String[] {"Código de paquete", "Tipo de servicio", "Destinatario", "Total de envío", "Tipo de pago"}
    );
}
```