POS

MANUAL TÉCNICO

PROGRAMA REALIZADO POR EL ALUMNO ALLEN GIANKARLO ROMÁN VÁSQUEZ CARNET 202004745 PARA EL CURSO INTRODUCCIÓN A LA PROGRAMACIÓN DE COMPUTADORAS I DE LA ESCUELA DE CIENCIAS Y SISTEMAS DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD SAN CARLOS DE GUATEMALA

CONTENIDO

IDENTIFICACIÓN DEL DOCUMENTO

El presente documento constituye el manual técnico del proyecto “POS” para la empresa Blue Mall, en el cual se proporciona al lector los aspectos que se consideraron para su elaboración.

Es importante destacar que este manual no es un curso de aprendizaje de las funciones de programación utilizadas para el desarrollo del programa, sino es una herramienta que provee los aspectos a conocer para la forma correcta de operación y aplicación de este.

Programa realizado por el alumno Allen Giankarlo Román Vásquez carné 202004745 para el curso Introducción a la Programación de Computadoras 1 de la escuela de ciencias y sistemas de la Facultad de Ingeniería de la Universidad san Carlos de Guatemala

LUGAR, FECHA Y RESPONSABLES DE LA ELABORACIÓN

El programa se elaboró en Huehuetenango para la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, durante las últimas dos semanas del mes de agosto del año 2021 durante el segundo semestre de este y el programa fue realizado por el alumno Allen Giankarlo Román Vásquez carné 202004745 para el curso Introducción a la Programación de Computadoras 1 de la escuela de ciencias y sistemas de la Facultad de Ingeniería de la Universidad san Carlos de Guatemala

OBJETIVOS Y ALCANCES DEL SISTEMA

Proveer de una herramienta de trabajo a administrador y vendedores para poder gestionar de una mejor eficaz y eficiente sus ventas.

Disponer de una interfaz libre y sencilla de utilizar para una empresa en la que puedan almacenar datos importantes minimizando el riesgo de pérdida de información.

Gestionar las ventas por sucursal de forma más sencilla, y que las mismas permitan proveer información separada o consolidada, y que la misma se encuentre de forma ordenada y oportuna para la toma de decisiones gerenciales.

Proveer información importante relativa a los clientes, para determinar el período de cobro promedio y si es acorde a las políticas de la empresa, asimismo, de los productos que se comercializan para determinar necesidades de compra al llegar a un mínimo establecido.

ESPECIFICACIÓN TÉCNICA (HARDWARE Y SOFTWARE)

Los requisitos para que el sistema pueda ser ejecutado adecuadamente son:

- Procesador: Intel Pentium III 800 MHz (800MHz Intel Pentium III u otro equivalente)
- RAM: 512 MB
- Espacio en disco: 750 MB
- Sistema Operativo: Windows 7, Windows XP, Windows Vista) Windows XP Profesional SP3/Vista SP1/Windows 7 Professional)
- Resolución gráfica: 1024 x 728
- Navegador de internet: Google Chrome, Microsoft Edge, Mozilla Firefox, Vivaldi u Opera
- Herramientas: Java y algún IDE que corra el mismo, en este caso se utilizo NetBeans 8.2.

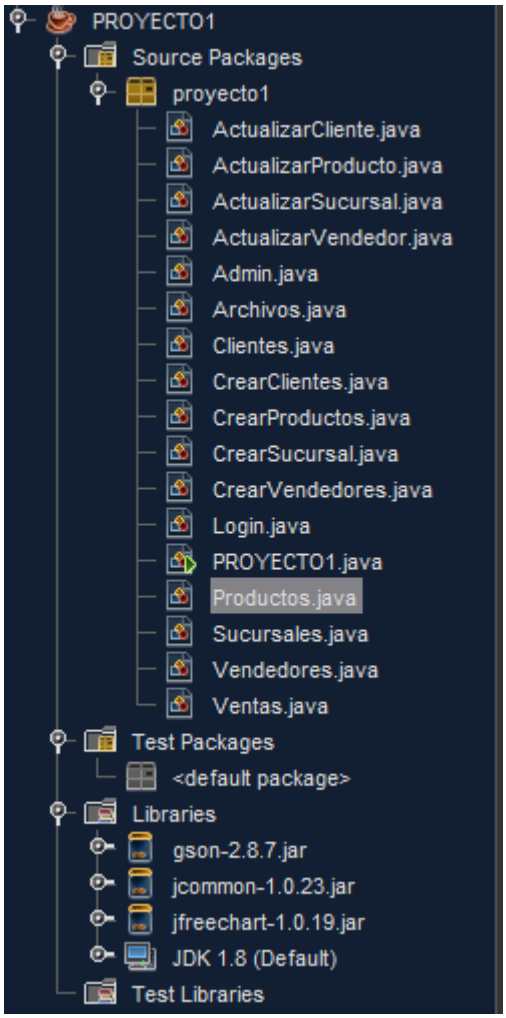
LÓGICA DEL PROGRAMA

Configuración del sistema

El sistema POS está desarrollado bajo un formato Java de nombre NetBeans en su versión 8.2, esto da la facilidad de estructurar el sistema de manera que se facilita el mantenimiento a dicha solución, a continuación, se describe la estructura básica del sistema y se enfatiza en los archivos y directorios relevantes para su configuración y adaptación.

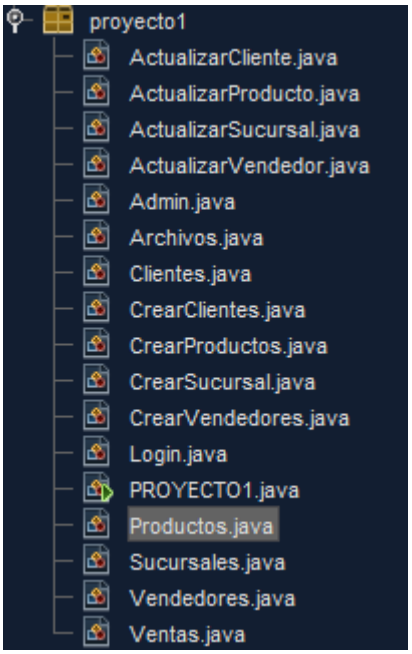
Estructura

El proyecto tiene la siguiente estructura.



Directorio

El proyecto tiene la siguiente estructura.



Package proyecto1: en el que se encuentran todas las clases utilizadas para la realización del proyecto.

1. **ActualizarCliente/ActualizarProducto/ActualizarSucursal/ActualizarVendedor:** En esta pestaña se edita los atributos según el objeto seleccionado.

```
this.setTitle("Actualizar Cliente");
this.setBounds(600,250,400,400);
this.setLocationRelativeTo(null);
this.getContentPane().setBackground(new Color(174, 214, 241));
this.setLayout(null);

JLabel titulo = new JLabel();
titulo.setText("Crear Actualizar Cliente");
titulo.setBounds(125, 25, 150, 25);
this.add(titulo);

JLabel label1 = new JLabel();
label1.setText("Código");
label1.setBounds(50, 60, 75, 25);
this.add(label1);

JLabel label2 = new JLabel();
label2.setText("Nombre");
label2.setBounds(50, 110, 75, 25);
this.add(label2);

JLabel label3 = new JLabel();
label3.setText("NIT");
label3.setBounds(50, 160, 75, 25);
this.add(label3);

JLabel label4 = new JLabel();
label4.setText("Correo");
label4.setBounds(50, 210, 75, 25);
this.add(label4);

JLabel label5 = new JLabel();
label5.setText("Genero");
label5.setBounds(50, 260, 75, 25);
this.add(label5);

ingreso1 = new JTextField();
ingreso1.setBounds(150,60,200,25);
this.add(ingreso1);

ingreso2 = new JTextField();
ingreso2.setBounds(150,110,200,25);
this.add(ingreso2);
```

```
ingreso2 = new JTextField();
ingreso2.setBounds(150,110,200,25);
this.add(ingreso2);

ingreso3 = new JTextField();
ingreso3.setBounds(150,160,200,25);
this.add(ingreso3);

ingreso4 = new JTextField();
ingreso4.setBounds(150,210,200,25);
this.add(ingreso4);

ingreso5 = new JTextField();
ingreso5.setBounds(150,260,200,25);
this.add(ingreso5);

agregar = new JButton();
agregar.setBounds(150, 310, 200, 25);
agregar.setText("Actualizar");
agregar.addActionListener(this);
this.add(agregar);
```

Esta porción de código genera la interfaz gráfica utilizada.

```
    @Override
    public void actionPerformed(ActionEvent ae) {
                int codigo = 0;
                 String codigo1 = ingreso1.getText();
                 codigo = Integer.parseInt(codigo1);
                 String nombre = ingreso2.getText();
                 String nit = ingreso3.getText();
                 String correo = ingreso4.getText();
                 String genero = ingreso5.getText();

                 Clientes nuevo = new Clientes(codigo, nombre, nit, correo, genero);

                 PROYECTO1.clientes[Admin.S] = nuevo;

            this.dispose();
            Admin.tablaclientes();
    }
}
```

Y esta última lee lo que se ingresa en cada JTextField y actualiza los datos, luego regresa al Módulo de ventas.

2. **Admin:** En esta clase se encuentra el módulo de administración, así como todas sus funcionalidades.

```
this.setTitle("Administración");
this.setBounds(600, 250, 1200, 900);
this.setLocationRelativeTo(null);
this.getContentPane().setBackground(new Color(174, 214, 241));
this.setLayout(null);

JTabbedPane pestañas = new JTabbedPane();
pestañas.setBounds(0, 80, 1400, 850);
this.add(pestañas);

JLabel titulo = new JLabel();
titulo.setText("Módulo de Administración");
titulo.setBounds(300, 20, 1000, 50);
titulo.setFont(new Font("Century Gothic", 1, 50));
this.add(titulo);
```

Configuración del título y las pestañas.

PARA CADA PANEL YA SEA SUCURSALES/CLIENTES/PRODUCTOS/VENDEDORES.

```
panel1 = new JPanel();
panel1.setBackground(new Color(52, 159, 219));
pestañas.add("Sucursales", panel1);
panel1.setLayout(null);
crear1 = new JButton();
crear1.setText("Crear");
crear1.setBounds(700, 50, 170, 50);
crear1.addActionListener(this);
panel1.add(crear1);
cargal = new JButton();
cargal.setText("Carga Masiva");
cargal.setBounds(900, 50, 170, 50);
cargal.addActionListener(this);
panel1.add(cargal);
```

Configuración del panel.

```
actualizar1 = new JButton();
actualizar1.setText("Actualizar");
actualizar1.setBounds(700, 120, 170, 50);

actualizar1.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        try {
            S = tablasucursal.getSelectedRow();
            ActualizarSucursal ventana = new ActualizarSucursal();
            ventana.setVisible(true);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "Seleccione una Sucursal.");
        }
    }
});
panel1.add(actualizar1);
```

Botón de actualizar que selecciona una fila y con getSelectedRow() y lo guarda en una variable S, y esto lleva al método de Actualizar en el que en base a la variable S se editan los atributos.

```
eliminar1 = new JButton();
eliminar1.setText("Eliminar");
eliminar1.setBounds(900, 120, 170, 50);
eliminar1.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        try {
            int S = tablasucursal.getSelectedRow();
            PROYECTO1.sucursales[S] = null;
            tablasucursal();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "Seleccione una Sucursal.");
        }
    }
});
panel1.add(eliminar1);
```

Botón de eliminar que selecciona una fila y la guarda en la variable S y en base a S hace nula la posición en el arreglo de objetos.

```
exportar1 = new JButton();
exportar1.setText("Exportar Listado a PDF");
exportar1.setBounds(700, 190, 370, 50);
exportar1.addActionListener(this);
panel1.add(exportar1);
tabla1 = new JTextField();
tabla1.setBounds(50, 50, 550, 675);
panel1.add(tabla1);
```

Tabla 1 es un TextField así cada vez que se imprime la tabla de sucursales se puede limpiar.

PARA CADA TABLA YA SEA SUCURSALES/CLIENTES/PRODUCTOS/VENDEDORES.

```
public static void tablasucursal() {
    tabla1.removeAll();
    String[] columnas = {"Codigo", "Nombre", "Dirección", "Correo", "Teléfono"};
    Object[][] datos = new Object[50][5];

    for (int i = 0; i < PROYECTO1.sucursales.length; i++) {
        if (PROYECTO1.sucursales[i] != null) {

            datos[i][0] = PROYECTO1.sucursales[i].getCodigo();
            datos[i][1] = PROYECTO1.sucursales[i].getNombre();
            datos[i][2] = PROYECTO1.sucursales[i].getDireccion();
            datos[i][3] = PROYECTO1.sucursales[i].getCorreo();
            datos[i][4] = PROYECTO1.sucursales[i].getTelefono();

        }

    }
    tablasucursal = new JTable(datos, columnas);
    JScrollPane scroll = new JScrollPane(tablasucursal);
    scroll.setBounds(0, 0, 550, 675);
    scroll.setVisible(true);
    tabla1.add(scroll);
}
```

Se limpia de primero el JTextField correspondiente, y se crear un arreglo de columnas, con los nombres de las columnas y uno de datos en lo que iría todos los datos del arreglo de objetos correspondiente, para luego

ingresarlo a un ciclo for en el que se ingresan uno a uno los objetos, para luego imprimirlo en el JTextField correspondiente agregando un scroll para la movilidad.

PARA GRÁFICA DE PRODUCTOS/VENDEDORES

```
public static void graficaproductos() {
    grafica2.removeAll();
    DefaultCategoryDataset barras = new DefaultCategoryDataset();
    int i = 0;
    for (int j = 0; j < PROYECTO1.productos.length; j++) {
        if (PROYECTO1.productos[j] != null) {
            barras.setValue(PROYECTO1.productos[j].getCantidad(), PROYECTO1.productos[j].getNombre(), PROYECTO1.productos[j].getNombre());
            i = j + 1;
            break;
        }
    }

    for (int j = i; j < PROYECTO1.productos.length; j++) {
        if (PROYECTO1.productos[j] != null) {
            barras.setValue(PROYECTO1.productos[j].getCantidad(), PROYECTO1.productos[j].getNombre(), PROYECTO1.productos[j].getNombre());
            i = j + 1;
            break;
        }
    }

    for (int j = i; j < PROYECTO1.productos.length; j++) {
        if (PROYECTO1.productos[j] != null) {
            barras.setValue(PROYECTO1.productos[j].getCantidad(), PROYECTO1.productos[j].getNombre(), PROYECTO1.productos[j].getNombre());
            break;
        }
    }

    JFreeChart barra1 = ChartFactory.createBarChart3D("Top 3 - Productos con más disponibilidad", "Producto", "Cantidad", barras, PlotOrientation.VERTICAL, true, true, false);
    ChartPanel barra2 = new ChartPanel(barra1);

    barra2.setBounds(0, 0, 370, 375);
    grafica2.add(barra2);

}
```

Se crea una barra y luego se ingresa a un ciclo for para corroborar que haya algún objeto en su posición y sino iterar hasta encontrar uno y así sucesivamente con los siguientes tres, para luego ingresar lo que requiere la table y su impresión.

```
public static void graficaclientes() {
    grafica3.removeAll();
    DefaultPieDataset datasetPie = new DefaultPieDataset();
    int hombre = 0;
    int mujer = 0;

    for (int i = 0; i < PROYECTO1.clientes.length; i++) {
        if (PROYECTO1.clientes[i] != null) {
            if (PROYECTO1.clientes[i].getGenero().equals("f")) {
                mujer++;
            }
        }
    }

    for (int i = 0; i < PROYECTO1.clientes.length; i++) {
        if (PROYECTO1.clientes[i] != null) {
            if (PROYECTO1.clientes[i].getGenero().equals("m")) {
                hombre++;
            }
        }
    }

    datasetPie.setValue("Femenino", new Integer(mujer));
    datasetPie.setValue("Masculino", new Integer(hombre));

    JFreeChart Pie = ChartFactory.createPieChart3D("Género de vendedores", datasetPie, true, true, false);
    ChartPanel piel = new ChartPanel(Pie);

    piel.setBounds(0, 0, 370, 375);
    grafica3.add(piel);
}
```

Gráfica de clientes que en base al ingreso realiza un conteo ya sea de hombre o mujer, para luego meterlo en una gráfica de Pie y agregarlo al JTextField correspondiente.

```
if (ae.getSource() == cerrar) {
    this.dispose();
    Login ventana = new Login();
    ventana.setVisible(true);
    Archivos.serialize("Vendedores.bin", PROYECTO1.vendedores);
    Archivos.serialize("Clientes.bin", PROYECTO1.clientes);
    Archivos.serialize("Productos.bin", PROYECTO1.productos);
    Archivos.serialize("Sucursales.bin", PROYECTO1.sucursales);
}
```

Botón de cerrar que ciera la ventana y llama al Login, esto serializando.

PARA CADA PESTAÑA

```
    if (ae.getSource() == cargal) {
        JFileChooser elegir = new JFileChooser();
        elegir.showOpenDialog(this);
        File archivo = elegir.getSelectedFile();
        String ruta = archivo.getAbsolutePath();
        Archivos.CargarSucursales(ruta);
        tablasucursal();
    }
    if (ae.getSource() == exportarl) {
        System.out.println("hola");
        Archivos.utilJTablePrint(tablasucursal, "Sucursales", "", rootPaneCheckingEnabled)
    }
    if (ae.getSource() == crearl) {
        CrearSucursal ventana = new CrearSucursal();
        ventana.setVisible(true);
        tablasucursal();
    }
```

Botón de carga que realiza la carga masiva, llamando al método correspondiente.

Botón que exporta a pdf llamando el método correspondiente.

Botón de crear que crear un nuevo objeto y llama al método.

3. Archivos

PARA CADA MÉTODO DE CARGA

```java
public static void CargarSucursales(String ruta) {
    String content = getContentOfFile(ruta);

    JsonParser parser = new JsonParser();
    JsonArray arreglo = parser.parse(content).getAsJsonArray();
    int contador = 0;
    do {
        for (int i = 0; i < arreglo.size(); i++) {
            contador += 1;
            JsonObject objeto = arreglo.get(i).getAsJsonObject();

            int codigo = objeto.get("codigo").getAsInt();
            String nombre = objeto.get("nombre").getAsString();
            String direccion = objeto.get("direccion").getAsString();
            String correo = objeto.get("correo").getAsString();
            int telefono = objeto.get("telefono").getAsInt();

            Sucursales nuevo = new Sucursales(codigo, nombre, direccion, correo, telefono);

            PROYECTO1.sucursales[i] = nuevo;
        }
    } while (contador <= 50);
}
```

En el que se ingresa una ruta la cual viene de otro procedimiento en este se encuentra un archivo Json, en el que al recorrer el for se llena el objeto esto mientres el do while sea menor al contador correspondiente porque no se puede exceder de cierta cantidad de objetos.

```java
public static String getContentOfFile(String pathname) {
    File archivo = null;
    FileReader fr = null;
    BufferedReader br = null;

    try {
        // Apertura del fichero y creacion de BufferedReader para poder
        // hacer una lectura comoda (disponer del metodo readLine()).
        archivo = new File(pathname);
        fr = new FileReader(archivo);
        br = new BufferedReader(fr);
        // Lectura del fichero
        String content = "";
        String linea;
        while ((linea = br.readLine()) != null) {
            content += linea + "\n";
        }
        return content;
    } catch (FileNotFoundException fnfe) {
        System.err.println("No se encontró el archivo. Inténtelo de nuevo");
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (null != fr) {
                fr.close();
            }
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return "";
}
```

Método en el cual se leerá el archivo y se abre el fichero para poder realizar la lectura.

```java
public static void utilJTablePrint(JTable jTable, String header, String footer, boolean showPrintDialog) {
    boolean fitWidth = true;
    boolean interactive = true;
    // (Definimos el modo de impresión)
    JTable.PrintMode mode = fitWidth ? JTable.PrintMode.FIT_WIDTH : JTable.PrintMode.NORMAL;
    try {
        // Print the table (Imprimo la tabla)
        boolean complete = jTable.print(mode,
                new MessageFormat(header),
                new MessageFormat(footer),
                showPrintDialog,
                null,
                interactive);
        if (complete) {
            // Mostramos el mensaje de impresión existosa
            JOptionPane.showMessageDialog(jTable,
                    "Print complete (Impresión completa)",
                    "Print result (Resultado de la impresión)",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            // Mostramos un mensaje indicando que la impresión fue cancelada
            JOptionPane.showMessageDialog(jTable,
                    "Print canceled (Impresión cancelada)",
                    "Print result (Resultado de la impresión)",
                    JOptionPane.WARNING_MESSAGE);
        }
    } catch (PrinterException pe) {
        JOptionPane.showMessageDialog(jTable,
                "Print fail (Fallo de impresión): " + pe.getMessage(),
                "Print result (Resultado de la impresión)",
                JOptionPane.ERROR_MESSAGE);
    }
}
```

Método para imprimir en pdf la tabla seleccionada, en el que trae del botón exportar la tabla solicitada, su título, pie de paágina y un boolean para imprimir cuadros diálogo.

```java
public static void serialize(String pathname, Object object) {
    // Serializar un objeto
    try {
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream(pathname));
        objectOutputStream.writeObject(object);
        objectOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static Object deserialize(String pathname) {
    // Leer un objeto serializado
    try {
        ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream(pathname));
        Object data = objectInputStream.readObject();
        objectInputStream.close();
        return data;
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
    return null;
}
```

Métodos para realizar la serealización y deserealización del programa, saliendo e ingresando con la ruta indicada, así como el objeto.

```java
public static void BurbujaDesc1(Productos[] arr) {
    int n = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] != null) {
            n++;
        }
    }
    Productos temp;
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < (n - i); j++) {
            if (arr[j - 1].getCantidad() < arr[j].getCantidad() && arr[j].getCantidad() != 0 && arr[j - 1].getCantidad() != 0) { //Solo se realiza el cambio de signo
                temp = arr[j - 1];
                arr[j - 1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

public static void BurbujaDesc2(Vendedores[] arr) {
    int n = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] != null) {
            n++;
        }
    }
    Vendedores temp;
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < (n - i); j++) {
            if (arr[j - 1].getVentas() < arr[j].getVentas() && arr[j].getVentas() != 0 && arr[j - 1].getVentas() != 0) { //Solo se realiza el cambio de signo
                temp = arr[j - 1];
                arr[j - 1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Ordenamientos de burbuja para poder realizar los gráficos de barra.

4. CLIENTES/SUCURSALES/PRODUCTOS/VENDEDORES

```java
public Clientes(int codigo, String nombre, String nit, String correo, String genero ){
    this.codigo= codigo;
    this.nombre=nombre;
    this.nit= nit;
    this.correo= correo;
    this.genero=genero;


}

public int getCodigo() {
    return codigo;
}

public void setCodigo(int codigo) {
    this.codigo = codigo;
}

public String getNombre() {
    return nombre;
}

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

public String getNit() {
    return nit;
}

public void setNit(String nit) {
    this.nit = nit;
}

public String getCorreo() {
    return correo;
}

public void setCorreo(String correo) {
    this.correo = correo;
}

public String getGenero() {
    return genero;
}

public void setGenero(String genero) {
    this.genero = genero;
}
```

En estas clases se realizan la declaración de objetos, así como los getters and seters.

5. Crear(Clientes/Productos/Vendedores/Sucursales)

```
public CrearClientes(){
    this.setTitle("Crear Cliente");
    this.setBounds(600,250,400,400);
    this.setLocationRelativeTo(null);
    this.getContentPane().setBackground(new Color(174, 214, 241));
    this.setLayout(null);

    JLabel titulo = new JLabel();
    titulo.setText("Crear Nuevo Cliente");
    titulo.setBounds(125, 25, 150, 25);
    this.add(titulo);

    JLabel label1 = new JLabel();
    label1.setText("Código");
    label1.setBounds(50, 60, 75, 25);
    this.add(label1);

    JLabel label2 = new JLabel();
    label2.setText("Nombre");
    label2.setBounds(50, 110, 75, 25);
    this.add(label2);

    JLabel label3 = new JLabel();
    label3.setText("NIT");
    label3.setBounds(50, 160, 75, 25);
    this.add(label3);

    JLabel label4 = new JLabel();
    label4.setText("Correo");
    label4.setBounds(50, 210, 75, 25);
    this.add(label4);

    JLabel label5 = new JLabel();
    label5.setText("Genero");
    label5.setBounds(50, 260, 75, 25);
    this.add(label5);

    ingreso1 = new JTextField();
    ingreso1.setBounds(150,60,200,25);
    this.add(ingreso1);

    ingreso2 = new JTextField();
    ingreso2.setBounds(150,110,200,25);
    this.add(ingreso2);
```

```
ingreso3 = new JTextField();
ingreso3.setBounds(150,160,200,25);
this.add(ingreso3);

ingreso4 = new JTextField();
ingreso4.setBounds(150,210,200,25);
this.add(ingreso4);

ingreso5 = new JTextField();
ingreso5.setBounds(150,260,200,25);
this.add(ingreso5);

agregar = new JButton();
agregar.setBounds(150, 310, 200, 25);
agregar.setText("Agregar");
agregar.addActionListener(this);
this.add(agregar);
```

Interfaz para cada clase de crear.

```
if(ae.getSource()==agregar){

  for (int i = 0; i < PROYECTO1.clientes.length; i++) {
      if(PROYECTO1.clientes[i]==null){
        int codigo = 0;
        String codigo1 = ingreso1.getText();
        codigo = Integer.parseInt(codigo1);
        String nombre = ingreso2.getText();
        String nit = ingreso3.getText();
        String correo = ingreso4.getText();
        String genero = ingreso5.getText();

        Clientes nuevo = new Clientes(codigo, nombre, nit, correo, genero);

        PROYECTO1.clientes[i] = nuevo;
        break;
           }
    }
    this.dispose();
    Admin.tablaclientes();
}
```

Para cada clase de crear, se lee lo ingresado en JTextField.

6. Login

```java
public Login() {
    this.setTitle("Modulo de Autentificación");
    this.setBounds(600, 250, 400, 250);
    this.setLocationRelativeTo(null);
    this.getContentPane().setBackground(new Color(174, 214, 241));
    this.setLayout(null);

    JLabel pos = new JLabel();
    pos.setText("POS");
    pos.setBounds(200, 25, 25, 25);
    this.add(pos);

    JLabel label1 = new JLabel();
    label1.setText("Código");
    label1.setBounds(50, 60, 75, 25);
    this.add(label1);

    JLabel label2 = new JLabel();
    label2.setText("Contraseña");
    label2.setBounds(50, 110, 75, 25);
    this.add(label2);

    ingreso1 = new JTextField();
    ingreso1.setBounds(150, 60, 200, 25);
    this.add(ingreso1);

    ingreso2 = new JPasswordField();
    ingreso2.setBounds(150, 110, 200, 25);
    this.add(ingreso2);

    inicio = new JButton();
    inicio.setText("Iniciar Sesión");
    inicio.setBounds(150, 160, 200, 25);
    inicio.addActionListener(this);
    this.add(inicio);
}
```

Interfaz del Login

```java
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == inicio) {
        String usuario = "";
        String contrasenia = "";
        System.out.println("Se presiono el botón");
        contrasenia = ingreso2.getText();
        usuario = ingreso1.getText();

        if (ingreso1.getText().equals("admin") && ingreso2.getText().equals("admin")) {
            System.out.println("User y password correctas");
            this.dispose();
            Admin ventanal = new Admin();
            ventanal.setVisible(true);
        } else if (usuario.equals("") && contrasenia.equals("")) {
            JOptionPane.showMessageDialog(null, "Ingrese un usuario");
        } else if (PROYECTO1.vendedores != null) {

            int usuario1 = Integer.parseInt(usuario);
            for (int i = 0; i < PROYECTO1.vendedores.length; i++) {
                if (usuario1 == PROYECTO1.vendedores[i].getCodigo() && contrasenia.equals(PROYECTO1.vendedores[i].getPassword()) && PROYECTO1.vendedores[i] != null) {
                    nombre = PROYECTO1.vendedores[i].getNombre();
                    this.dispose();
                    Ventas ventanal = new Ventas();
                    ventanal.setVisible(true);
                    break;
                }
            }
        } else {
            JOptionPane.showMessageDialog(null, "asdf");
        }

    }
}
```

Lee que se ingresa en los JTextField y en base a eso ingresa al módulo correspondiente.

7. PROYECTO1

```
public static void main(String[] args) {
    if (new File("Vendedores.bin").exists()) {
        PROYECTO1.vendedores = (Vendedores[]) Archivos.deserialize("Vendedores.bin");
    }
    if (new File("Clientes.bin").exists()) {
        PROYECTO1.clientes = (Clientes[]) Archivos.deserialize("Clientes.bin");
    }
    if (new File("Productos.bin").exists()) {
        PROYECTO1.productos = (Productos[]) Archivos.deserialize("Productos.bin");
    }
    if (new File("Sucursales.bin").exists()) {
        PROYECTO1.sucursales = (Sucursales[]) Archivos.deserialize("Sucursales.bin");
    }
    Login ventana = new Login();
    ventana.setVisible(true);
}
```

En la ejecución corrobora si hay archivos serealizados y luego pasa a desplegar el Login.

Libraries: en las que se encuentran las tres librerías utilizadas, entre ellas la gson-2.8.7.jar que realiza manejo de los archivos jar, jcommon-1.0.23.jar y jfreechart-1.0.19.jar que genera las gráficas.