Prueba de Caja Blanca

"Sistema de administración de pedidos comedor Sabor Manaba"

Integrantes:

Andrade Uriel

Matias Jean

Plua Thomas

Fecha: 2025-02-09

INDICE

Primer requisito

Historia de Usuario	
Número: REQ001	Usuario: Alex Gonzales
Nombre Historia: Menú principal del sistema	
Prioridad: alta	
Programador Responsable: Uriel Adrade	
Descripción: Mantener un control del menú mediante las opciones, Pedidos, Clientes y Reportes	
Validación: Que se activen botones del evento correcto (ingresar pedido, ingresar cliente, reporte)	

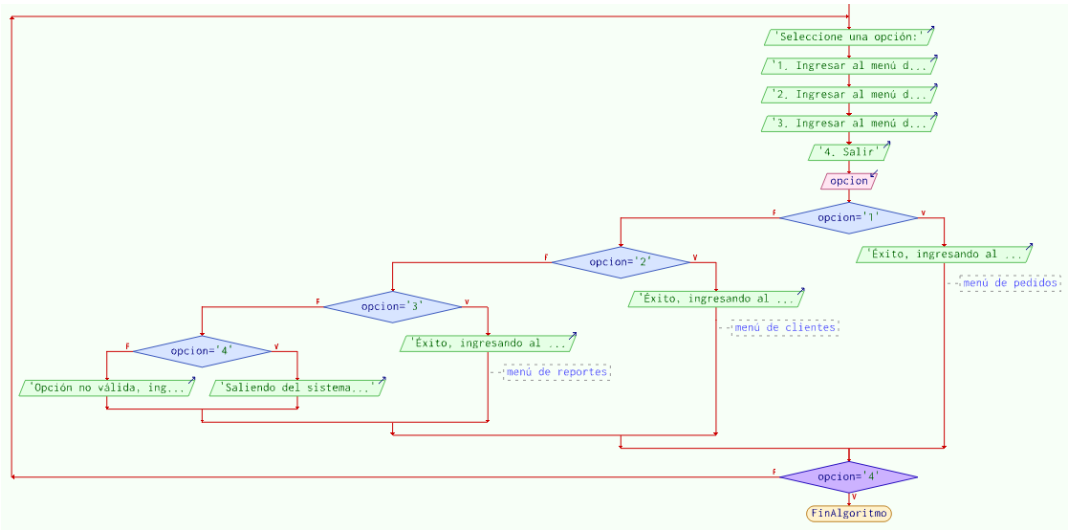
1. Código fuente

```
if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
    if (this.Opcion.getText().equals("1")) { //entrada a Pedidos
        //ENTRADA AL MENU PEDIDOS
        JOptionPane.showMessageDialog(this, "Exito, ingresando al menu pedidos", "Opcion Valida", JOptionPane.INFORMATION_MESSAGE);
        getMenuPedidos().setVisible(true);
        this.setVisible(false);
        getOpcion().setText("");
    } else if (this.Opcion.getText().equals("2")) { //entrada a Clientes
        //ENTRADA AL MENU CLIENTES
        JOptionPane.showMessageDialog(this, "Exito, ingresando al menu clientes", "Opcion Valida", JOptionPane.INFORMATION_MESSAGE);
        getMenuClientes().setVisible(true);
        this.setVisible(false);
        getOpcion().setText("");
    } else if (this.Opcion.getText().equals("3")) { //entrada a Reportes
        //ENTRADA AL MENU REPORTES
        JOptionPane.showMessageDialog(this, "Exito, ingresando al menu reportes", "Opcion Valida", JOptionPane.INFORMATION_MESSAGE);
        getMenuReporte().setVisible(true);
        this.setVisible(false);
        getOpcion().setText("");
       } else if (this.Opcion.getText().equals("4")) {
           JOptionPane.showMessageDialog(this, "Saliendo del sistema", "Salir", JOptionPane.CLOSED_OPTION);
           System.exit(0); //salir del programa
       } else {
           //MENSAJE DE OPCION NO VALIDA DEL IF
           JOptionPane.showMessageDialog(this, "Opcion no valida, ingrese una opcion entre el 1 y 4 ", "Opcion Invalida", JOptionPane.WARNING_MESSAGE);
           getOpcion().setText("");
       }
   }
```

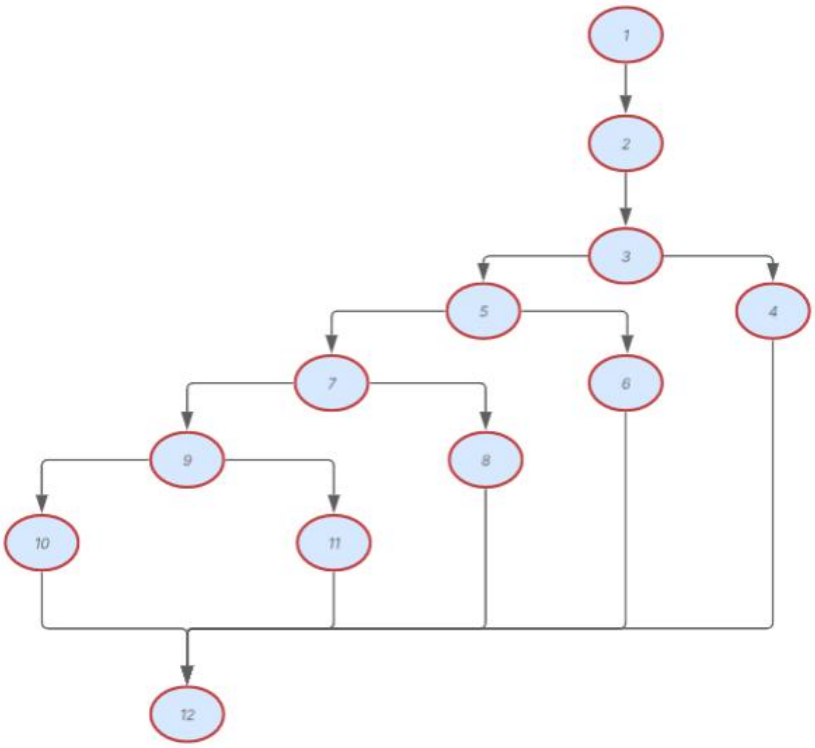
2. Algoritmo

```
Proceso Menu_Principal
    Definir opcion Como Cadena

    Repetir
        Escribir "Seleccione una opción:"
        Escribir "1. Ingresar al menú de pedidos"
        Escribir "2. Ingresar al menú de clientes"
        Escribir "3. Ingresar al menú de reportes"
        Escribir "4. Salir"
        Leer opcion

        Si opcion = "1" Entonces
            Escribir "Éxito, ingresando al menú de pedidos"
            //menú de pedidos
        Sino
            Si opcion = "2" Entonces
                Escribir "Éxito, ingresando al menú de clientes"
                // menú de clientes
            Sino
                Si opcion = "3" Entonces
                    Escribir "Éxito, ingresando al menú de reportes"
                    //menú de reportes
                Sino
                    Si opcion = "4" Entonces
                        Escribir "Saliendo del sistema..."
                    Sino
                        Escribir "Opción no válida, ingrese una opción entre 1 y 4"
                    FinSi
                FinSi
            FinSi
        FinSi
    Hasta Que opcion = "4"
FinProceso
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1. Rutas:

R1: 1,2,3,4,12
R2: 1,2,3,5,6,12
R3: 1,2,3,5,7,8,12
R4: 1,2,3,5,7,9,11,12
R5: 1,2,3,5,7,9,10,12

5.2. Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 4+1=**5**

V(G)= A- N +2
V(G)= 15-12+2= **5**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Segundo requisito

Historia de Usuario	
Número: REQ002	**Usuario:** Alex Gonzales
Nombre Historia: Ingreso de datos del pedido	
Prioridad: Alta	
Programador Responsable: Uriel Andrade	
Descripción: Se debe realizar el registro del pedido de algún cliente	
Validación: Si no se selecciona ningún platillo ofertado, si no se selecciona ningún método de pago.	

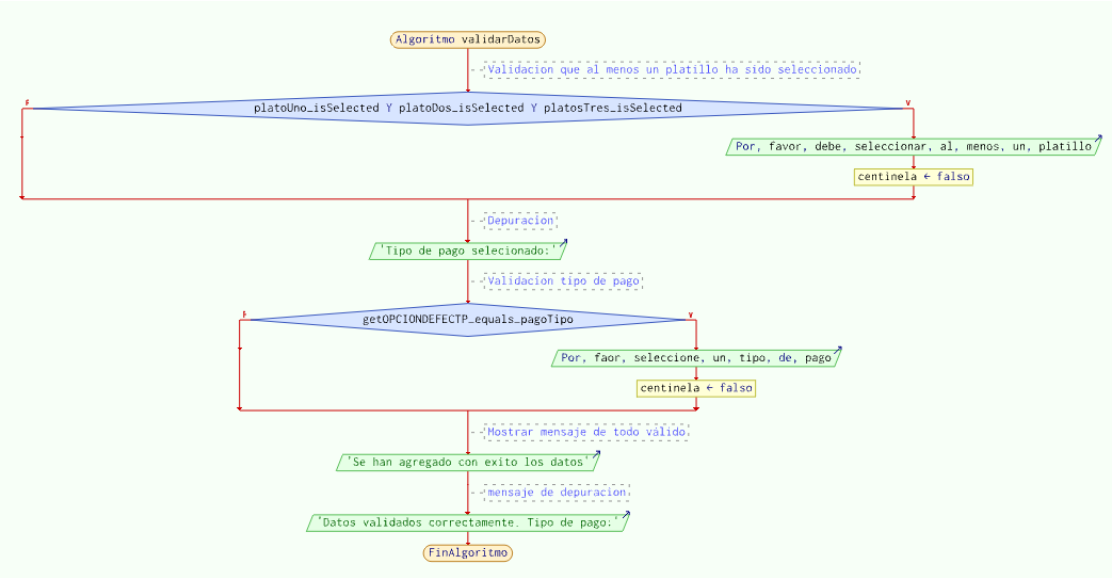
1. Código fuente

```java
public boolean validarDatos() {
    String pagoTipo = (String) this.getTipoPago().getSelectedItem();
    // Validación de que al menos un platillo ha sido seleccionado
    if (!this.platoUno.isSelected() && !this.platoDos.isSelected() && !this.platoTres.isSelected()) {
        mensaje("Por favor, debe seleccionar al menos un platillo.", "Advertencia");
        return false;
    }

    // Validación de tipo de pago
    System.out.println("Tipo de pago seleccionado: " + pagoTipo); // Depuración
    if (getOPCION_DEFECTO().equals(pagoTipo)) {
        mensaje("Por favor, seleccione un tipo de pago.", "Advertencia");
        return false;
    }

    // Mostrar mensaje de que todo es válido
    mensaje("Se han agregado con éxito los datos.", "Datos Agregados");

    // Mensaje de depuración
    System.out.println("Datos validados correctamente. Tipo de pago: " + pagoTipo);

    return true;
}
```

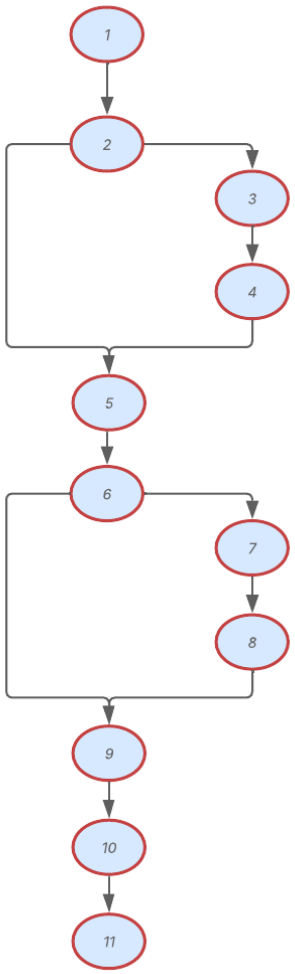
2. Algoritmo

```
Algoritmo validarDatos
    // Validacion que al menos un platillo ha sido seleccionado
    Si platoUno_isSelected ∧ platoDos_isSelected ∧ platosTres_isSelected Entonces
        Escribir Por favor, debe seleccionar al menos un platillo
        centinela ← falso
    FinSi
    // Depuracion
    Escribir "Tipo de pago selecionado:"
    // Validacion tipo de pago
    Si getOPCIONDEFECTP_equals_pagoTipo Entonces
        Escribir Por faor, seleccione un tipo de pago
        centinela ← falso
    FinSi
    // Mostrar mensaje de todo válido
    Escribir "Se han agregado con exito los datos"
    // mensaje de depuracion
    Escribir "Datos validados correctamente. Tipo de pago:"
FinAlgoritmo
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1.Rutas:

R1: 1,2,3,4,5,6,7,8,9,10,11
R2: 1,2,5,6,7,8,9,10,11
R3: 1,2,3,4,5,6,9,10,11

5.2.Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 2+1=**3**

V(G)= A- N +2
V(G)= 12-11+2=**3**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Tercer requisito

Historia de Usuario	
Número: REQ003	**Usuario:** Alex Gonzales
Nombre Historia: Ingreso de datos del cliente	
Prioridad: Alta	
Programador Responsable: Uriel Andrade	
Descripción: Se debe realizar el registro del pedido de algún cliente	
Validación: Si no se selecciona ningún platillo ofertado, si no se selecciona ningún método de pago.	

1. Código fuente

```
    if (nombre.isEmpty()) {
        mostrarError("Por favor, complete el nombre del cliente.");
        return false;
    }
    if (!nombre.matches("^[\\p{L}\\s]+$")) {
        mostrarError("El nombre solo debe contener letras.");
        return false;
    }

    // VALIDACIÓN DEL APELLIDO: no vacío y solo letras
    if (apellido.isEmpty()) {
        mostrarError("Por favor, complete el apellido del cliente.");
        return false;
    }
    if (!apellido.matches("^[\\p{L}\\s]+$")) {
        mostrarError("El apellido solo debe contener letras.");
        return false;
}

// VALIDACIÓN DEL TIPO DE IDENTIFICACIÓN
if (getOPCION_DEFECTO().equals(tipoIdentificacion)) {
    mostrarError("Por favor, seleccione un tipo de identificación válido.");
    return false;
}

// VALIDACIÓN DE LA IDENTIFICACIÓN SEGÚN EL TIPO
if (tipoIdentificacion.equals("Cedula")) {
    // La cédula debe tener 10 dígitos y ser válida
    if (!identificacion.matches("\\d{10}")) {
        mostrarError("La cédula debe tener 10 dígitos.");
        return false;
    }
    if (!validarCedula(identificacion)) {
        mostrarError("La cédula no es válida.");
        return false;
    }
} else if (tipoIdentificacion.equals("RUC")) {
    // El RUC debe tener 13 dígitos
    if (!identificacion.matches("\\d{13}")) {
```

```
            mostrarError("El RUC debe tener 13 dígitos.");
            return false;
        }
        // Para RUC de persona natural, los primeros 10 dígitos deben ser una cédula válida y los últimos
        String cedulaPart = identificacion.substring(0, 10);
        String sufijo = identificacion.substring(10);
        if (!validarCedula(cedulaPart) || !sufijo.equals("001")) {
            mostrarError("El RUC no es válido.");
            return false;
        }
    } else if (tipoIdentificacion.equals("Pasaporte")) {
        // Por ejemplo, se puede exigir que el pasaporte sea alfanumérico y tenga entre 6 y 9 caracteres
        if (!identificacion.matches("[A-Za-z0-9]{6,9}")) {
            mostrarError("El pasaporte debe ser alfanumérico y tener entre 6 y 9 caracteres.");
            return false;
        }
    }

    // VALIDACIÓN DE LA DIRECCIÓN: no vacía
    if (direccion.isEmpty()) {
        mostrarError("Por favor, complete la dirección del cliente.");
        return false;
    }
// VALIDACIÓN DEL TELÉFONO: no vacío y debe ser un número válido de Ecuador (ej. 09xxxxxxxx, 10 dígitos)
if (telefono.isEmpty()) {
    mostrarError("Por favor, complete el teléfono del cliente.");
    return false;
}
if (!telefono.matches("^09\\d{8}$")) {
    mostrarError("Por favor, ingrese un teléfono válido de Ecuador (debe comenzar con '09' y tener 10 dígitos).");
    return false;
}

// VALIDACIÓN DEL CORREO: no vacío y formato válido
if (correo.isEmpty()) {
    mostrarError("Por favor, complete el correo del cliente.");
    return false;
}
if (!correo.matches("^[\\w.-]+@[\\w.-]+\\.[a-zA-Z]{2,}$")) {
    mostrarError("Por favor, ingrese un correo válido.");
    return false;
}
//</editor-fold>

return true;
```

2. Algoritmo

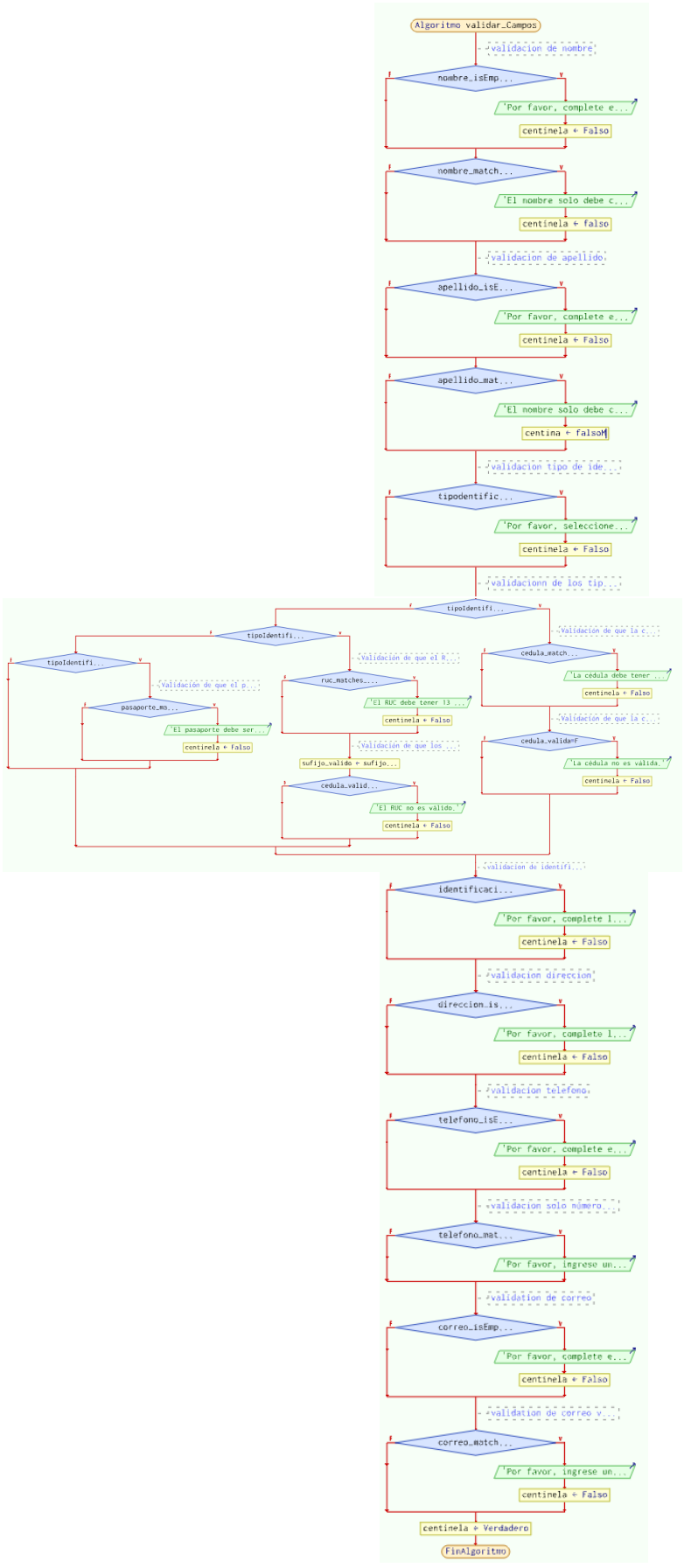
```
Algoritmo validar_Campos
    // validacion de nombre
    Si nombre_isEmpty = V Entonces
        Escribir 'Por favor, complete el nombre del cliente.'
        centinela ← Falso
    FinSi
    Si nombre_matches ≠ V Entonces
        Escribir 'El nombre solo debe contener letras.'
    FinSi

    // validacion de apellido
    Si apellido_isEmpty = V Entonces
        Escribir 'Por favor, complete el apellido del cliente.'
        centinela ← Falso
    FinSi
    Si apellido_matches ≠ V Entonces
        Escribir 'El nombre solo debe contener letras.'
    FinSi

    // validacion tipo de identificacion (combo_box)
    Si tipodentificación_DEFECTO = V Entonces
        Escribir 'Por favor, seleccione un tipo de identificación válido.'
        centinela ← Falso
    FinSi

    //validacionn de los tipos de identificacion
    Si tipoIdentificacion = "Cedula" Entonces
        // Validación de que la cédula tenga 10 dígitos
        Si cedula_matches_digitos = F Entonces
            Escribir 'La cédula debe tener 10 dígitos.'
            centinela ← Falso
        FinSi

```

```
        // Validación de que la cédula sea válida
        Si cedula_valida = F Entonces
            Escribir 'La cédula no es válida.'
            centinela ← Falso
        FinSi
        Sino Si tipoIdentificacion = "RUC" Entonces
        // Validación de que el RUC tenga 13 dígitos

        Si ruc_matches_digitos = F Entonces
            Escribir 'El RUC debe tener 13 dígitos.'
            centinela ← Falso
        FinSi

        // Validación de que los primeros 10 dígitos sean una cédula válida y los últimos 3 sean "001"

        sufijo_valido ← sufijo = "001"
        Si cedula_valida = F O sufijo_valido = F Entonces
            Escribir 'El RUC no es válido.'
            centinela ← Falso
        FinSi

        Sino Si tipoIdentificacion = "Pasaporte" Entonces
                // Validación de que el pasaporte sea alfanumérico y tenga entre 6 y 9 caracteres
                Si pasaporte_matches_formato = F Entonces
                    Escribir 'El pasaporte debe ser alfanumérico y tener entre 6 y 9 caracteres.'
                    centinela ← Falso
                FinSi
            finsi
        FinSi
    FinSi
```

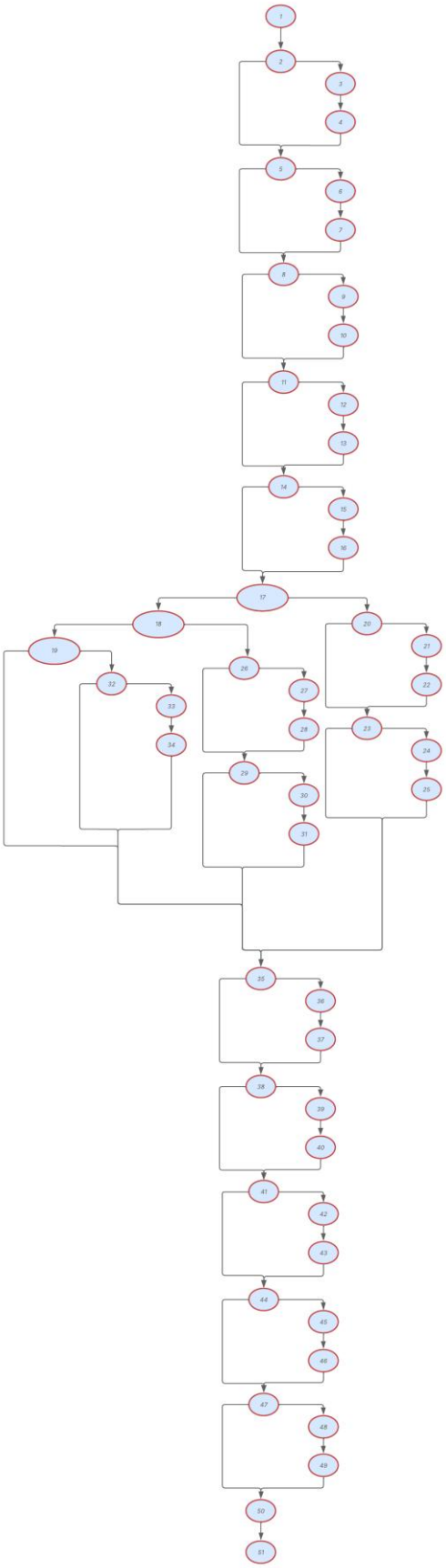
```
    // validacion de identificacion
    Si identificacion_isEmpty = V Entonces
        Escribir 'Por favor, complete la identificación del cliente.'
        centinela ← Falso
    FinSi

    // validacion direccion
    Si direccion_isEmpty = V Entonces
        Escribir 'Por favor, complete la dirección del cliente.'
        centinela ← Falso
    FinSi


    // validacion telefono
    Si telefono_isEmpty = V Entonces
        Escribir 'Por favor, complete el telefono del cliente.'
        centinela ← Falso
    FinSi

    // validacion solo números telefono y 10 digitos
    Si telefono_matches ≠ V Entonces
        Escribir 'Por favor, ingrese un teléfono válido de Ecuador (debe comenzar con 09 y tener 10 dígitos).'
    FinSi
```

```
    // validation de correo
    Si correo_isEmpty = V Entonces
        Escribir 'Por favor, complete el correo del cliente.'
        centinela ← Falso
    FinSi

    // validation de correo valido
    Si correo_matches = V Entonces
        Escribir 'Por favor, ingrese un correo válido.'
        centinela ← Falso
    FinSi

    centinela ← Verdadero
FinAlgoritmo
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1.Rutas:

R1: 1,2,3,51
R2: 1,2,4,51
R3: 1,5,6,51
R4: 1,5,7,51
R5: 1,8,51
R6: 1,9,10,11,51
R7: 1,9,10,12,51
R8:1,9,13,14,51
R9: 1,9,13,15,51
R10: 1,9,16,17,51
R11: 1,18,51
R12: 1,19,51
R13: 1,20,51
R14: 1,21,51
R15: 1,22,51
R16: 1,23,51
R17: 1,50,51

5.2.Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 18+1=**19**

V(G)= A- N +2
V(G)= 68-51+2= **19**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Cuarto requisito

Historia de Usuario	
Número: REQ004	**Usuario:** Alex Gonzales
Nombre Historia: Emisión de reportes	
Prioridad: Alta	
Programador Responsable: Uriel Andrade	
Descripción: Se deben mostrar los datos ya registrados	
Validación: Si no se han ingresado los datos de pedido y cliente con anterioridad.	

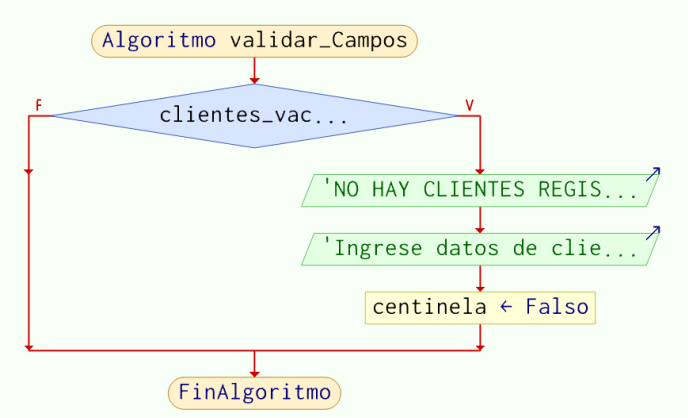
1. Código fuente

```
if (clientes.isEmpty()) {
    getjTextArea1().setText("No hay clientes registrados.".toUpperCase());
    JOptionPane.showMessageDialog(this, "Ingrese datos de cliente y pedido.", "Advertencia", JOptionPane.ERROR_MESSAGE);
    return;
}
```

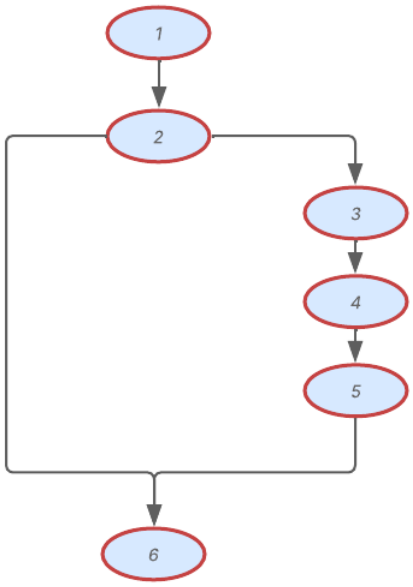
2. Algoritmo

```
1  Algoritmo validar_Campos
2      Si clientes_vacio = V Entonces
3          escribir "NO HAY CLIENTES REGISTRADOS."
4          escribir 'Ingrese datos de cliente y pedido'
5          centinela ← Falso
6      FinSi
7  FinAlgoritmo
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1.Rutas:

R1: 1,2,6
R2: 1,2,3,4,5,6

5.2.Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 1+1=**2**

V(G)= A- N +2
V(G)= 6-6+2= **2**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Quinto requisito

Historia de Usuario	
Número: REQ005	**Usuario:** Alex Gonzales
Nombre Historia: Eliminación de registro	
Prioridad: Alta	
Programador Responsable: Jean Matias	
Descripción: Se debe eliminar los reportes ya ingresados	
Validación: Confirmación de reporte eliminado	

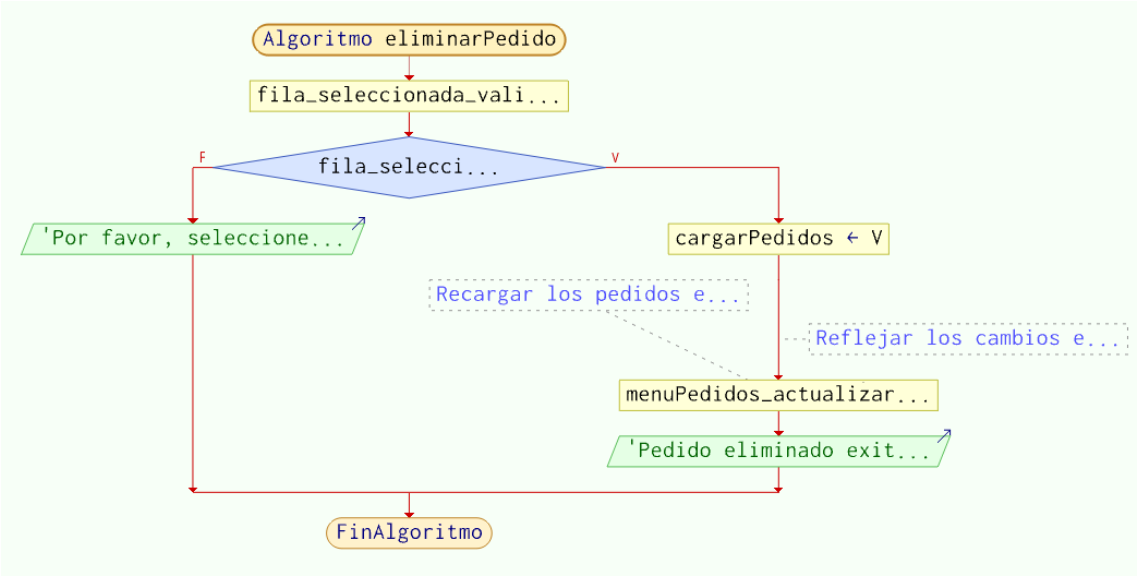
1. Código fuente

```
private void eliminarPedido() {
    int selectedRow = tablaPedidos.getSelectedRow();
    if (selectedRow != -1) {
        listaPedidos.remove(selectedRow); // Eliminar el pedido de la lista
        cargarPedidos(); // Recargar los pedidos en la tabla

        // Reflejar los cambios en MenuPedidos
        menuPedidos.actualizarDatosPedidos();

        JOptionPane.showMessageDialog(this, "Pedido eliminado exitosamente.", "Información", JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(this, "Por favor, seleccione un pedido para eliminar.", "Advertencia", JOptionPane.WARNING_MESSAGE);
    }
}
```

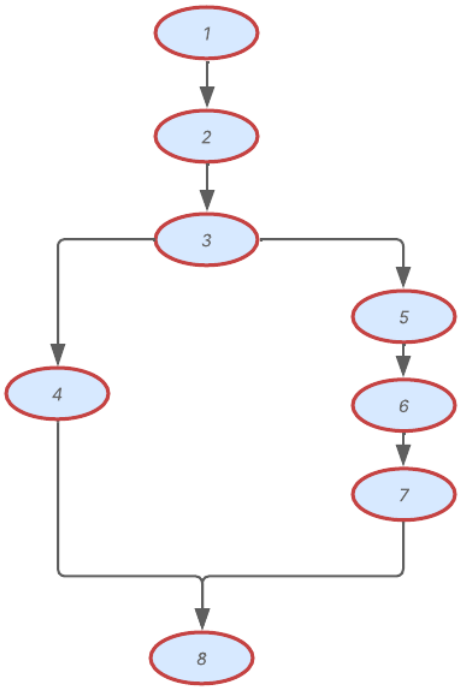
2. Algoritmo

```
1  Algoritmo  eliminarPedido
2  fila_seleccionada_valida ← selectedRow ≠ -1
3
4  Si fila_seleccionada_valida = V Entonces
5      cargarPedidos = V// Recargar los pedidos en la tabla
6
7      // Reflejar los cambios en MenuPedidos
8      menuPedidos_actualizarDatosPedidos = V
9
10     Escribir 'Pedido eliminado exitosamente.'
11 Sino
12     escribir 'Por favor, seleccione un pedido para eliminar'
13 FinSi
14 FinAlgoritmo
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1.Rutas:

R1: 1,2,3,4,8
R2: 1,2,3,5,6,7,8

5.2.Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 1+1=**2**

V(G)= A- N +2
V(G)= 8-8+2= **2**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Sexto requisito

Historia de Usuario	
Número: REQ006	**Usuario:** Alex Gonzales
Nombre Historia: Modificación de registro	
Prioridad: Alta	
Programador Responsable: Thomas Plua	
Descripción: Se deben mostrar los datos ya registrados	
Validación: Se visualiza en la tabla que los datos han sido modificados	

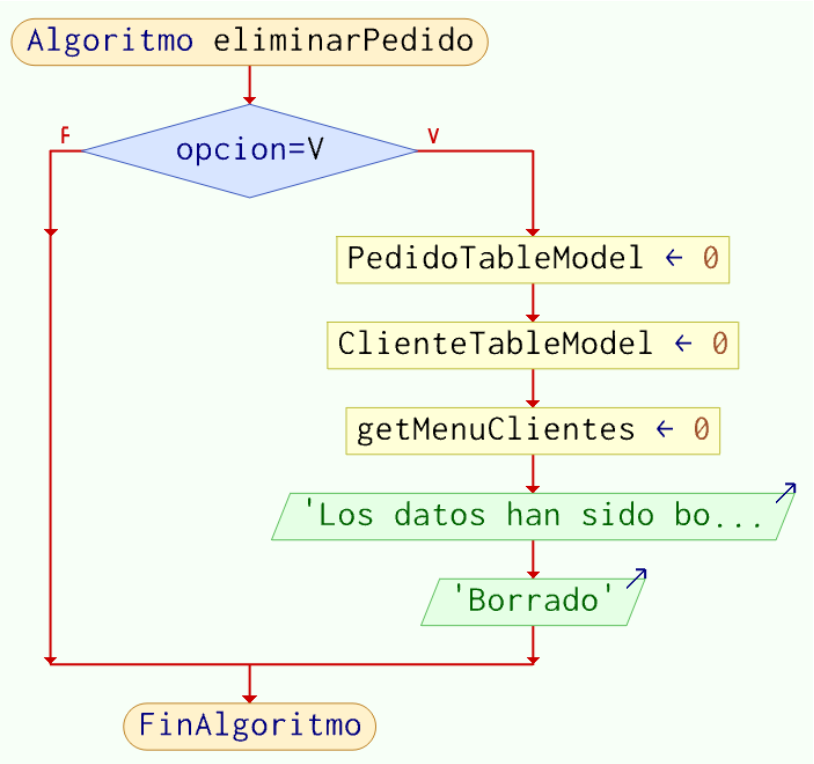
1. Código fuente

```
if (opcion == JOptionPane.YES_OPTION) {
    // Borrar datos de la tabla de pedidos asignándole un modelo vacío
    PedidoTableModel emptyPedidoModel = new PedidoTableModel(new ArrayList<>(), platos);
    tablaPedidos.setModel(emptyPedidoModel);

    // Borrar datos de la tabla de clientes asignándole un modelo vacío
    ClienteTableModel emptyClienteModel = new ClienteTableModel(new ArrayList<>());
    tablaClientes.setModel(emptyClienteModel);

    // Si además quieres borrar los datos de las listas subyacentes, puedes hacerlo:
    getMenuClientes().getClientes().clear();
    // Y para los pedidos, si tienes una lista, también:
    listaPedidos.clear();
    JOptionPane.showMessageDialog(this,
            "Los datos han sido borrados.",
            "Borrado",
            JOptionPane.INFORMATION_MESSAGE);
}
```

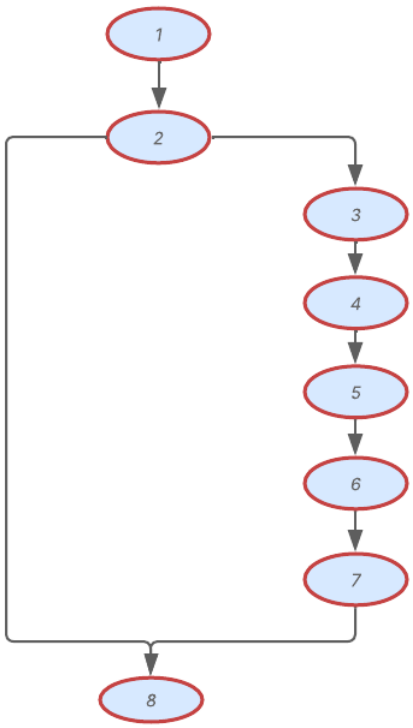
2. Algoritmo

```
1  Algoritmo  eliminarPedido
2      Si opcion = V Entonces
3          PedidoTableModel = 0
4          ClienteTableModel = 0
5          getMenuClientes = 0
6          escribir 'Los datos han sido borrados.'
7          escribir 'Borrado'
8      FinSi
9  FinAlgoritmo
```

3. Diagrama de flujo



4. Grafo de flujo



5. Identificación de rutas

5.1.Rutas:

R1: 1,2,8
R2: 1,2,3,4,5,6,7,8

5.2.Complejidad sistemática

V(G)= número de nodos predicados(decisiones)+1
V(G)= 1+1=**2**

V(G)= A- N +2
V(G)= 8-8+2= **2**

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos