

# Prueba de Caja Blanca

---

***“Desarrollo de una Plataforma para la Automatización  
de  
Procesos en la Empresa Gestión Integral en el Ámbito de  
Certificaciones ISO y BASC”***

**Integrantes:**

**Jhaldry Peñaherrera, José María Sandoval y Diego Pinto**

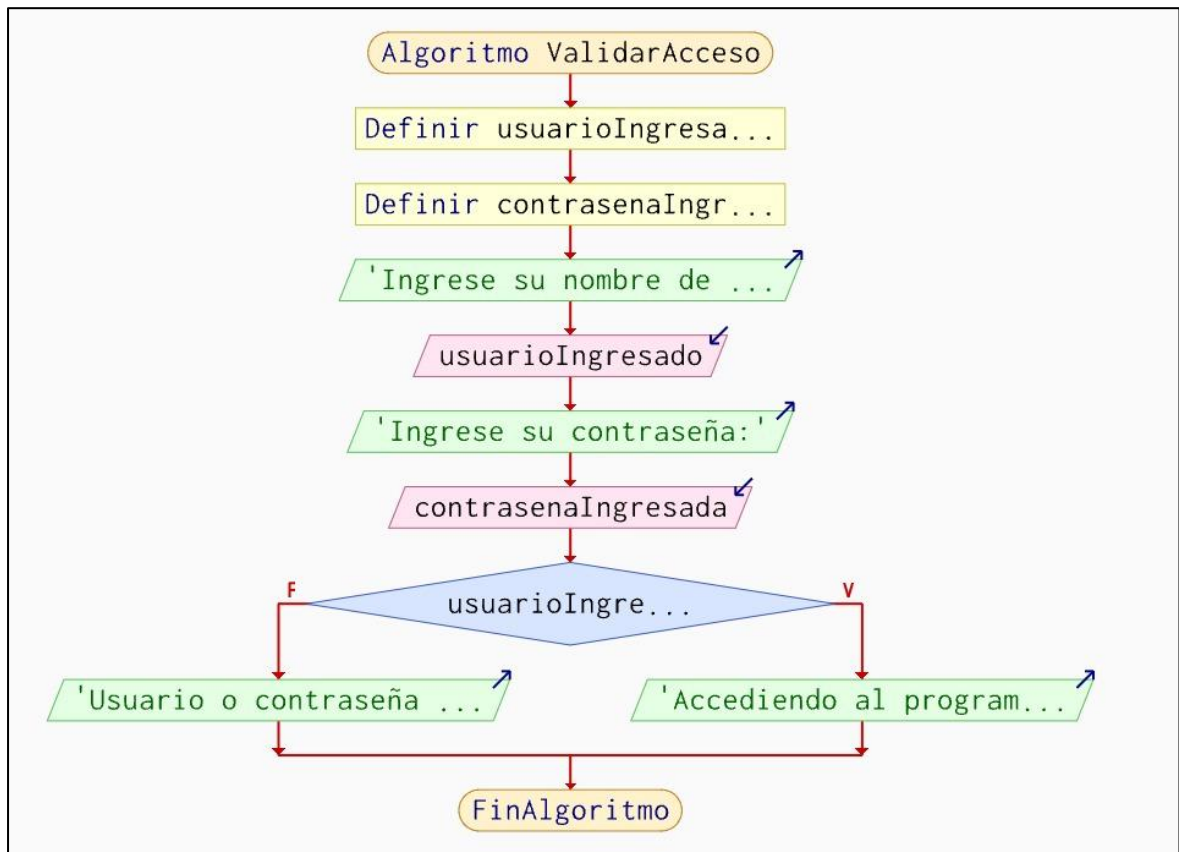
**Fecha 2025-07-09**

## Prueba caja blanca de: Inicio de sesión con Usuario y Contraseña

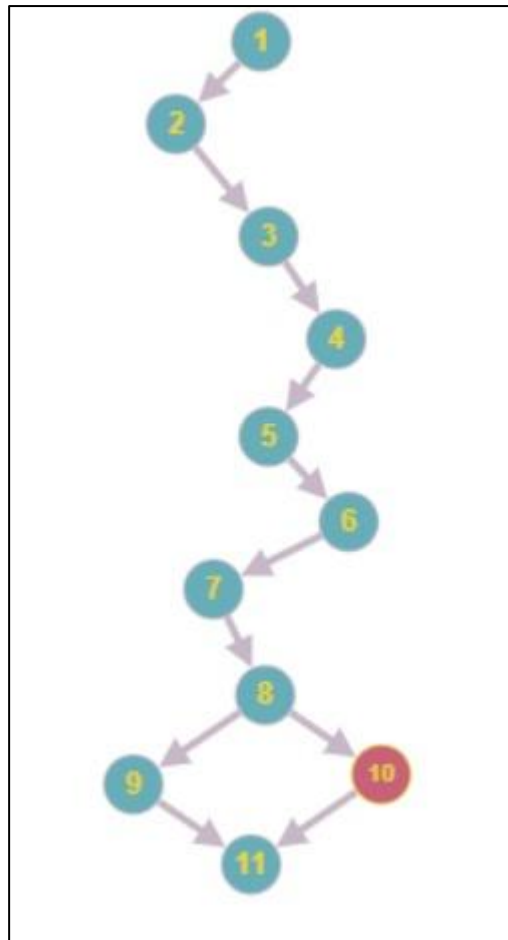
### 1. CÓDIGO FUENTE

```
private void verificarLogin(TextField txtUsuario, PasswordField txtContraseña, Label lblMensaje, Stage stage) {  
    String usuario = txtUsuario.getText();  
    String contraseña = txtContraseña.getText();  
    if (usuarios.containsKey(usuario) && usuarios.get(usuario)[0].equals(contraseña)) {  
        String rol = usuarios.get(usuario)[1];  
        mostrarPantallaPanel(stage, usuario, rol);  
    } else {  
        lblMensaje.setText(value:"Usuario o contraseña incorrectos");  
    }  
}
```

### 2. DIAGRAMA DE FLUJO (DF)



### 3. GRAFO DE FLUJO (GF)



### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino basico)

#### RUTAS

**R1:** 1,2,3,4,5,6,7,8,9,11

**R2:** 1,2,3,4,5,6,7,8,10,11

### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predicados (decisiones)} + 1$   
 $V(G) = 1 + 1$   
 $V(G) = 2$
- $V(G) = A - N + 2$   
 $V(G) = 11 - 11 + 2$   
 $V(G) = 2$

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

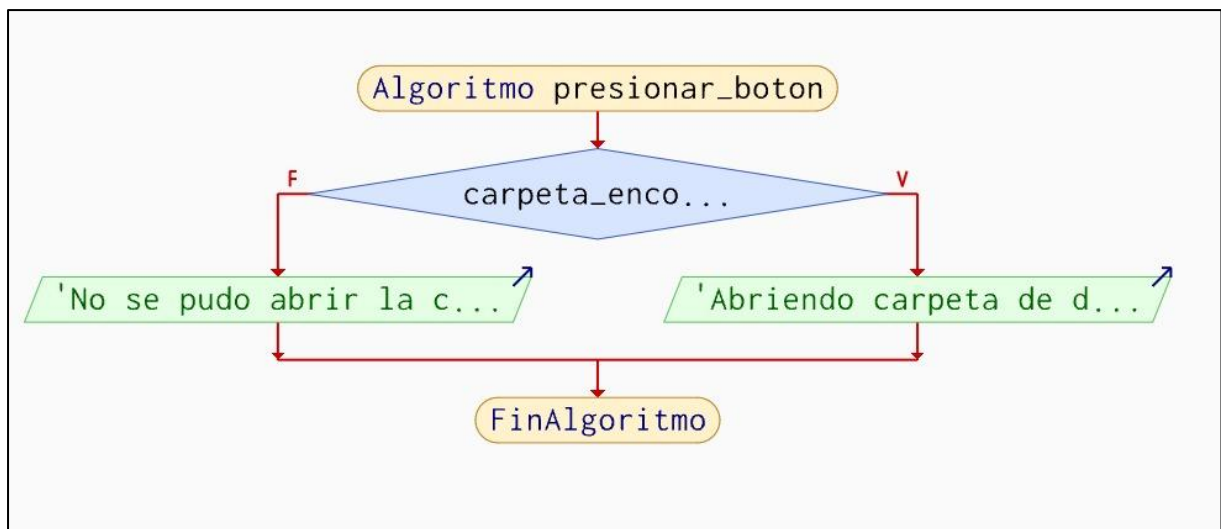
**N:** Número de nodos

**Prueba caja blanca de:** Enlace de mapa de procesos con documentos informativos.

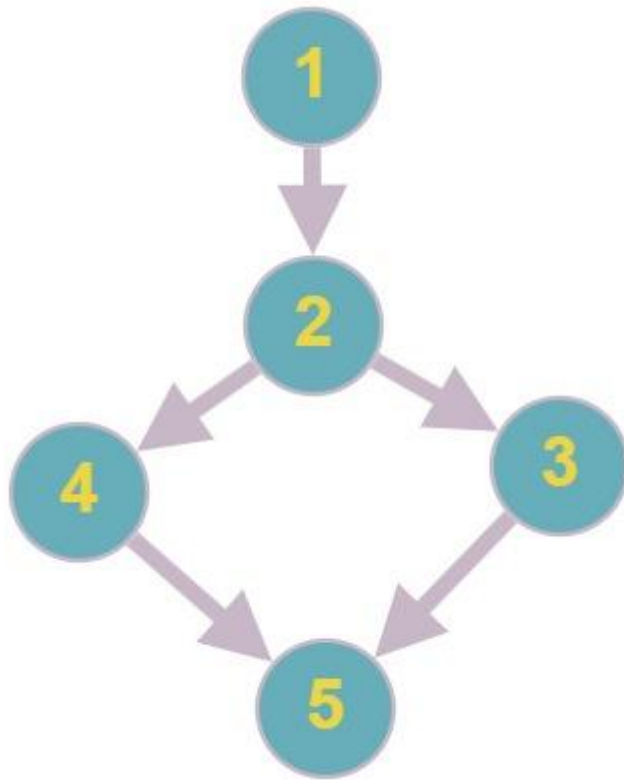
## 1. CÓDIGO FUENTE

```
btnFirmaContrato.setOnAction(_ -> {  
    try {  
        String folderPath = "C:\\Users\\santy\\OneDrive\\Documentos\\Carpeta documentos\\FIRM  
        java.awt.Desktop.getDesktop().open(new java.io.File(folderPath));  
    } catch (Exception ex) {  
        lblMensaje.setText(value:"No se pudo abrir la carpeta de documentos.");  
    }  
});
```

## 2. DIAGRAMA DE FLUJO (DF)



### 3. GRAFO DE FLUJO (GF)



### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

#### RUTAS

**R1:** 1,2,3,5

**R2:** 1,2,4,5

### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predcados (decisiones)} + 1$   
 $V(G) = 1 + 1$   
 $V(G) = 2$
- $V(G) = A - N + 2$   
 $V(G) = 5 - 5 + 2$   
 $V(G) = 2$

DONDE:

**P:** Número de nodos predcado

**A:** Número de aristas

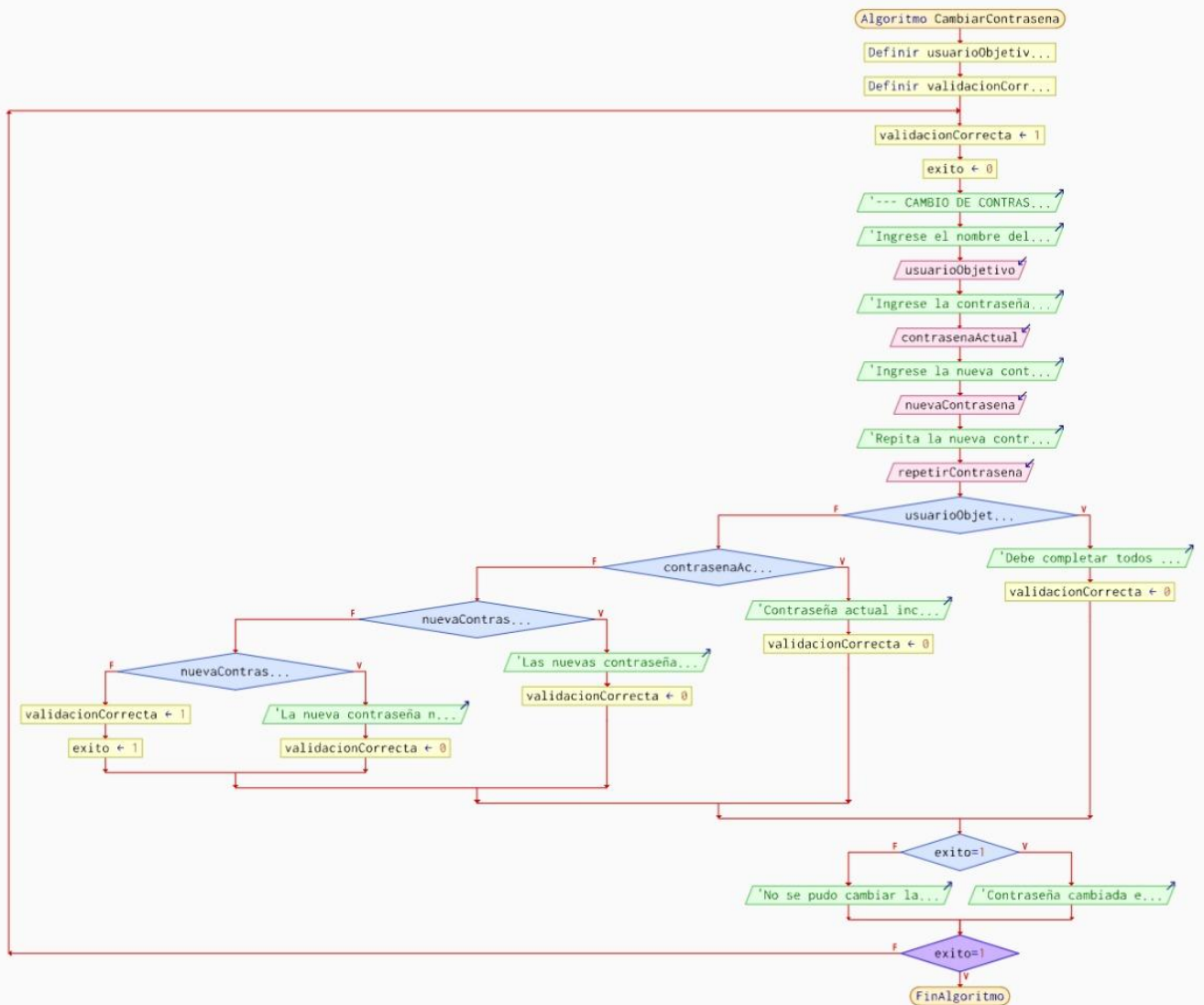
**N:** Número de nodos

## Prueba caja blanca de: Modificar las credenciales de los usuarios

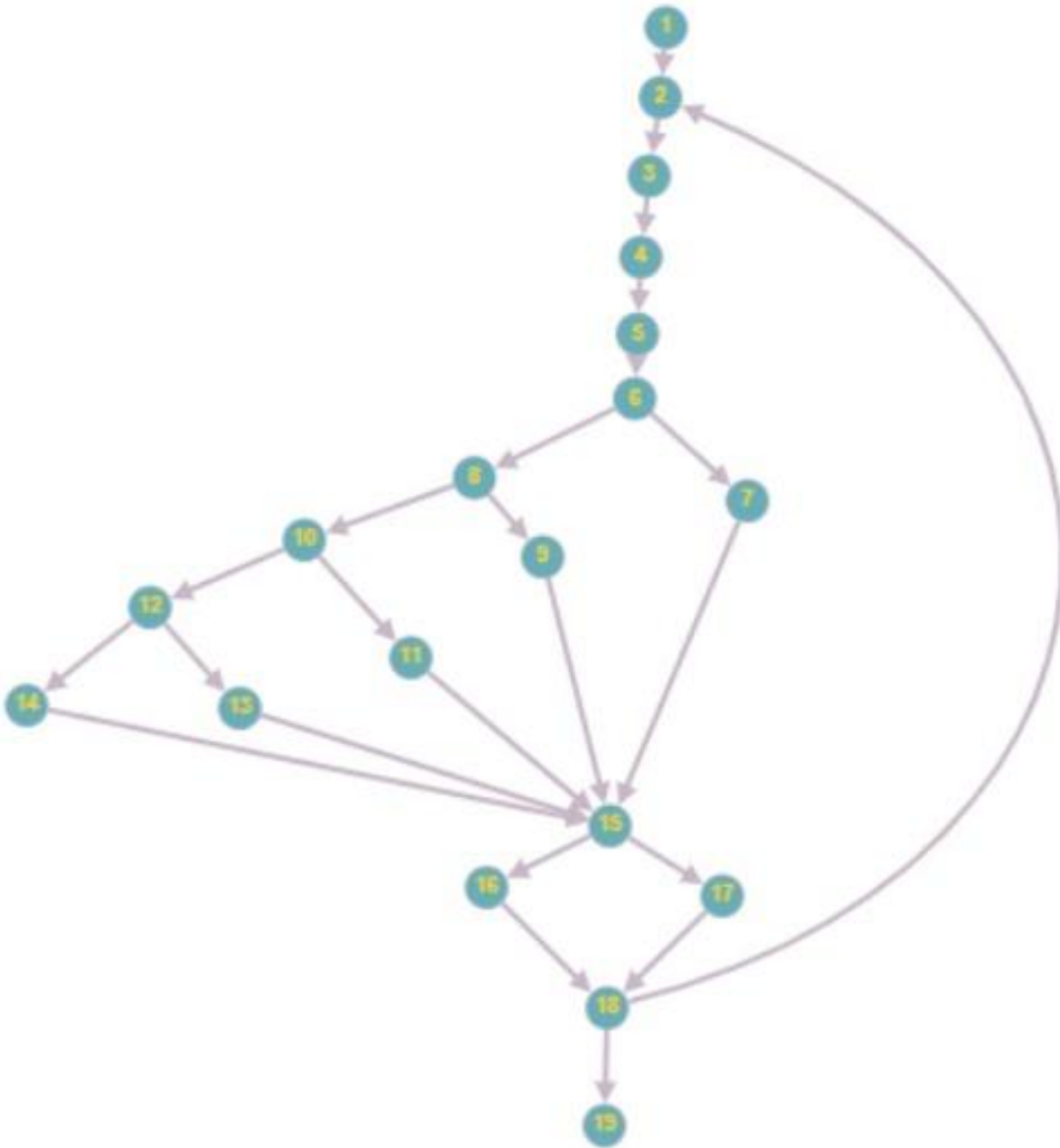
### 1. CÓDIGO FUENTE

```
boolean exito = false;
String usuarioObjetivo = comboUsuario.getValue();
String contrasenaActual = txtContrasenaActual.getText();
String nuevaContrasena = txtNuevaContrasena.getText();
String repetirContrasena = txtRepetirContrasena.getText();
if (usuarioObjetivo == null || usuarioObjetivo.isEmpty() || contrasenaActual == null || contrasenaActual.isEmpty() || nuevaContrasena == null || nuevaContrasena.isEmpty() || repetirContrasena == null || repetirContrasena.isEmpty()) {
    DialogUtils.mostrarAviso(formBox, mensaje:"Debe completar todos los campos.", esExito:false, () -> {});
    return;
}
// Solo el Tester puede cambiar la contraseña de otros usuarios, pero debe ingresar la contraseña actual del usuario objetivo
if (!usuarioService.verificarCredenciales(usuarioObjetivo, contrasenaActual)) {
    DialogUtils.mostrarAviso(formBox, mensaje:"Contraseña actual incorrecta.", esExito:false, () -> {});
    return;
}
if (!nuevaContrasena.equals(repetirContrasena)) {
    DialogUtils.mostrarAviso(formBox, mensaje:"Las nuevas contraseñas no coinciden.", esExito:false, () -> {});
    return;
}
if (nuevaContrasena.equals(contrasenaActual)) {
    DialogUtils.mostrarAviso(formBox, mensaje:"La nueva contraseña no puede ser igual a la actual.", esExito:false, () -> {});
    return;
}
usuarioService.cambiarCredenciales(usuarioObjetivo, nuevoUsuario:null, nuevaContrasena);
exito = true;
if (exito) {
    DialogUtils.mostrarAviso(formBox, mensaje:"Contraseña cambiada exitosamente.", esExito:true, () -> stageCambio.close());
} else {
    DialogUtils.mostrarAviso(formBox, mensaje:"No se pudo cambiar la contraseña.", esExito:false, () -> {});
}
}
```

## 2. DIAGRAMA DE FLUJO (DF)



### 3. GRAFO DE FLUJO (GF)





#### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

##### RUTAS

**R1:** 1,2,3,4,5,6,8,10,12,14,15,16,18,19

**R2:** No llegan al final

#### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predichos (decisiones)} + 1$   
 $V(G) = 5 + 1$   
 $V(G) = 6$
- $V(G) = A - N + 2$   
 $V(G) = 24 - 19 + 2$   
 $V(G) = 3$

DONDE:

**P:** Número de nodos predichos

**A:** Número de aristas

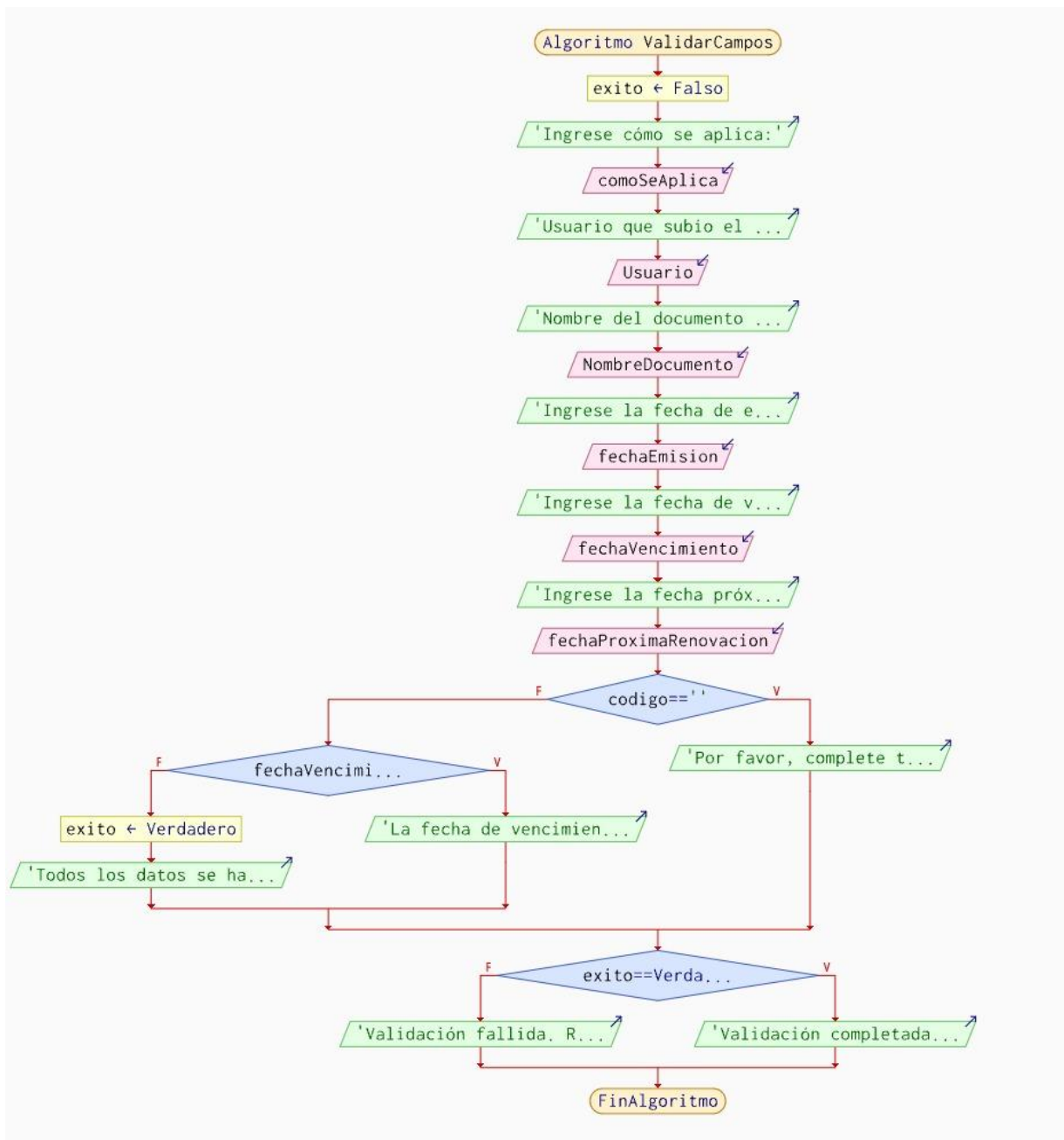
**N:** Número de nodos

## Prueba caja blanca de: La elaboración de la tabla de control de documentos

### 1. CÓDIGO FUENTE

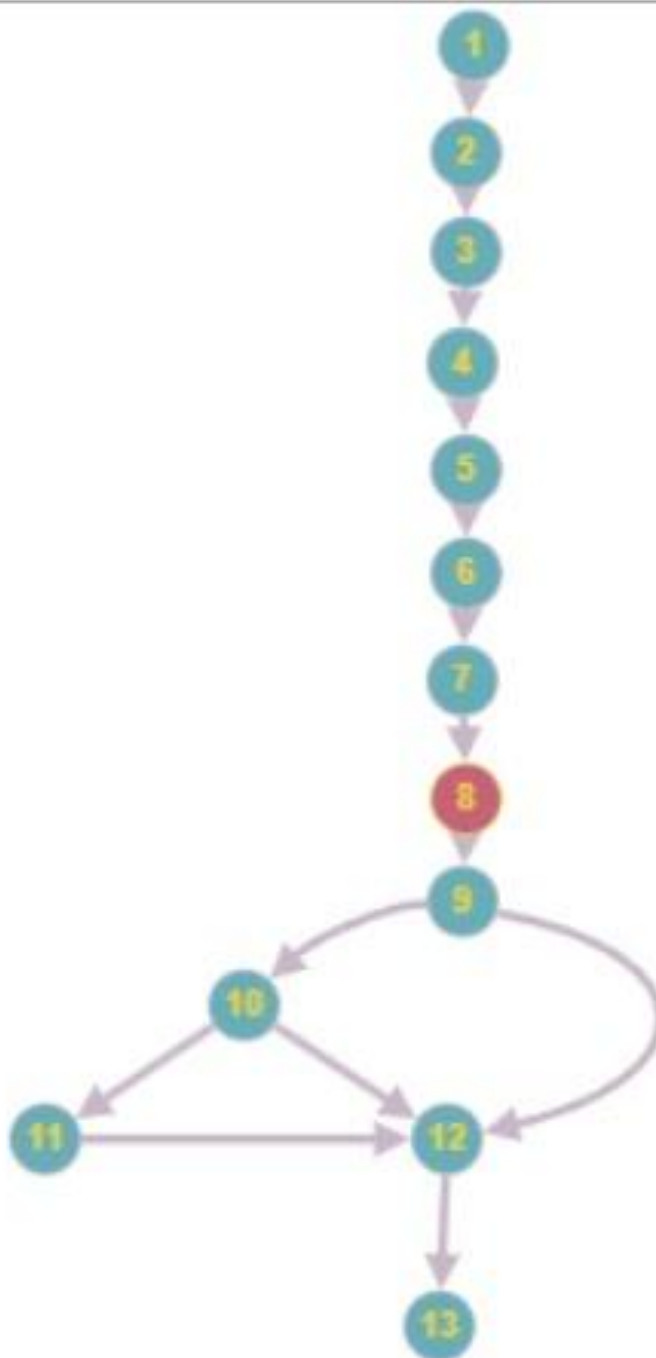
```
if (usuario.isEmpty() || nombreDocumento.isEmpty() || fechaEmision == null ||  
    (!chkNoVence.isSelected() && (fechaVencimientoStr == null || fechaProximaRenovacionStr == null))) {  
    DialogUtils.mostrarAviso(grid, mensaje:"Por favor, complete todos los campos correctamente. Si selecciona 'No vence', no debe ingresar fechas de vencimiento ni de  
    return;  
}  
  
// Validar que al menos se haya seleccionado un archivo  
if (archivosSeleccionados == null || archivosSeleccionados.isEmpty()) {  
    DialogUtils.mostrarAviso(grid, mensaje:"Debe adjuntar al menos un archivo para registrar el documento.", esExito:false, onSuccess:null);  
    return;  
}  
  
// Validación de fechas solo si no es "No vence"  
if (!chkNoVence.isSelected()) {  
    LocalDate fechaVencimiento = dtpFechaVencimiento.getValue();  
    LocalDate fechaProximaRenovacion = dtpFechaProximaRenovacion.getValue();  
    if (fechaVencimiento != null && fechaVencimiento.isBefore(fechaEmision)) {  
        DialogUtils.mostrarAviso(grid, mensaje:"La fecha de vencimiento no puede ser anterior a la fecha de emisión.", esExito:false, onSuccess:null);  
        return;  
    }  
    if (fechaProximaRenovacion != null) {  
        if (!fechaProximaRenovacion.isAfter(fechaEmision)) {  
            DialogUtils.mostrarAviso(grid, mensaje:"La fecha próxima de renovación debe ser mayor que la fecha de emisión.", esExito:false, onSuccess:null);  
            return;  
        }  
        if (fechaVencimiento != null && !fechaProximaRenovacion.isBefore(fechaVencimiento)) {  
            DialogUtils.mostrarAviso(grid, mensaje:"La fecha próxima de renovación debe ser menor que la fecha de vencimiento.", esExito:false, onSuccess:null);  
            return;  
        }  
    }  
}
```

## 2. DIAGRAMA DE FLUJO (DF)



### 3. GRAFO DE FLUJO (GF)

---



#### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

##### RUTAS

**R1:** 1,2,3,4,5,6,7,8,9,11,12,13

**R2:** 1,2,3,4,5,6,7,8,9,10,12,13

**R3:** 1,2,3,4,5,6,7,8,9,12,13,

#### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predcados (decisiones)} + 1$   
 $V(G) = 3 + 1$   
 $V(G) = 4$
- $V(G) = A - N + 2$   
 $V(G) = 14 - 13 + 2$   
 $V(G) = 3$

DONDE:

**P:** Número de nodos predcado

**A:** Número de aristas

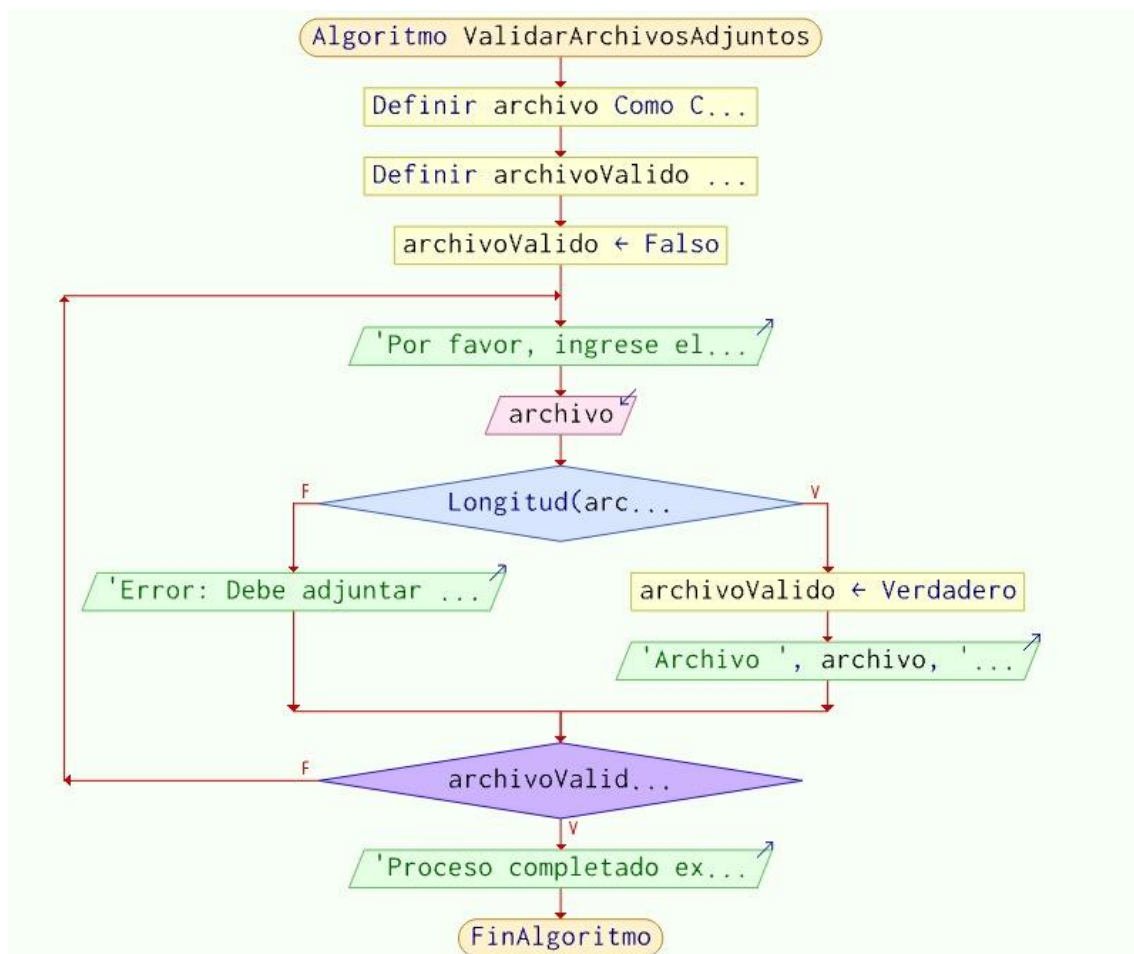
**N:** Número de nodos

## Prueba caja blanca de: Guardar la Información en el sistema

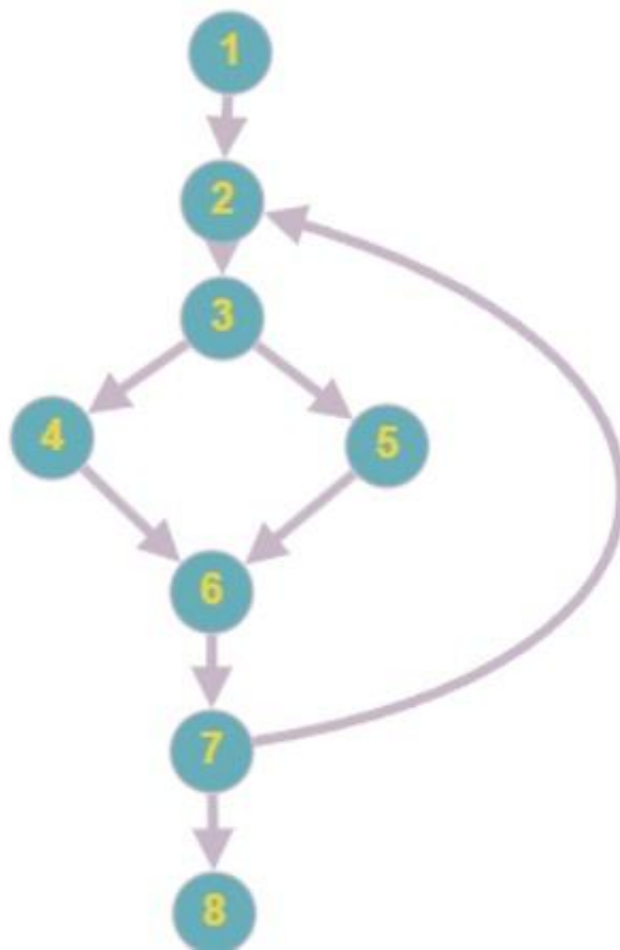
### 1. CÓDIGO FUENTE

```
// Validar que al menos se haya seleccionado un archivo
if (archivosSeleccionados == null || archivosSeleccionados.isEmpty()) {
    DialogUtils.mostrarAviso(grid, mensaje:"Debe adjuntar al menos un archivo para registrar el documento.", esExito:false, onSuccess:null);
    return;
}
```

### 2. DIAGRAMA DE FLUJO (DF)



### 3. GRAFO DE FLUJO (GF)



### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

#### RUTAS

**R1:** 1,2,3,4,6,7,8

**R2:** No llegan al final

### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predicados (decisiones)} + 1$   
 $V(G) = 2 + 1$   
 $V(G) = 3$
- $V(G) = A - N + 2$   
 $V(G) = 9 - 8 + 2$   
 $V(G) = 3$

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

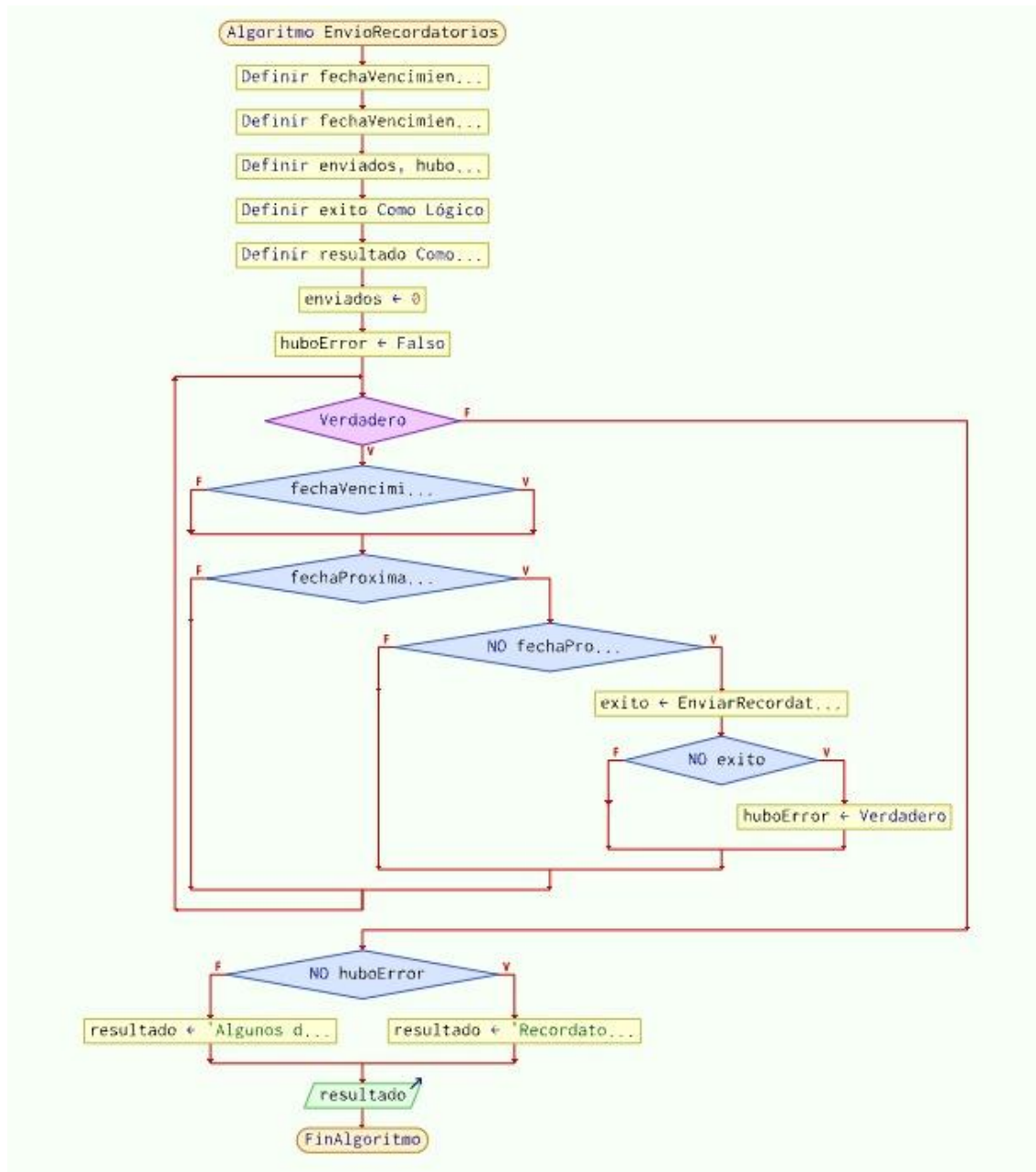
**N:** Número de nodos

## Prueba caja blanca de: Generar y enviar recordatorios

### 1. CÓDIGO FUENTE

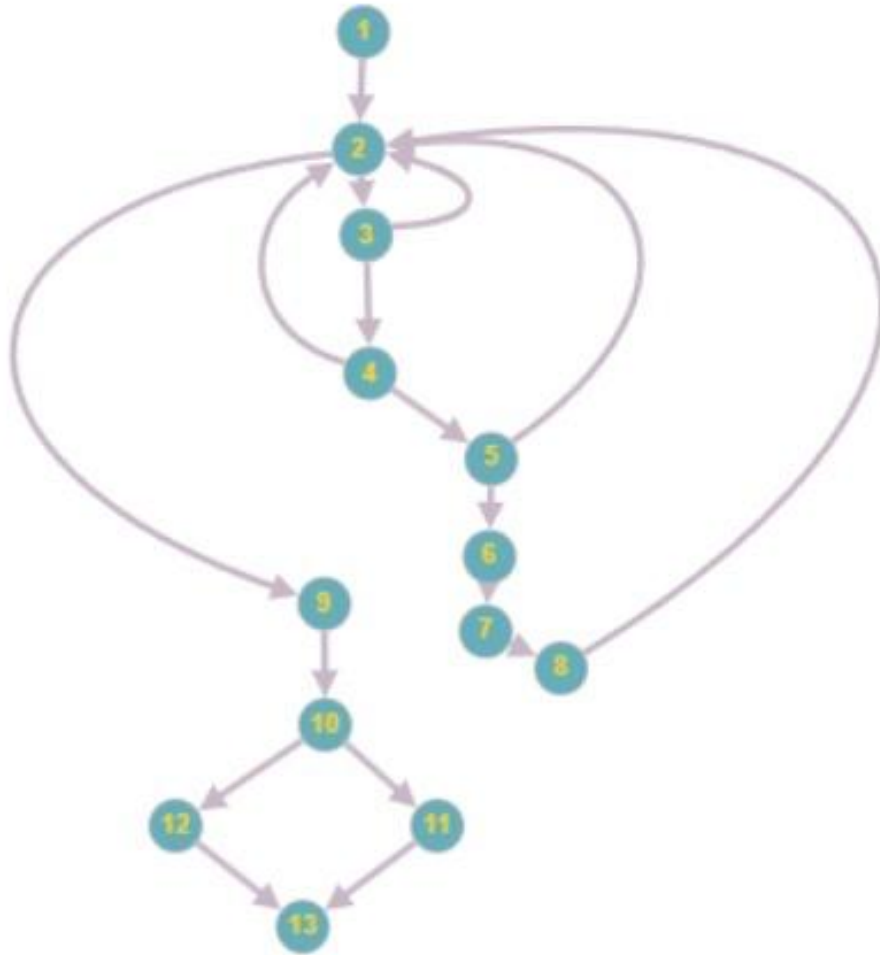
```
if ("NV".equals(fechaVencimientoObj) || "NV".equals(fechaProximaRenovacionObj)) {  
    continue;  
}  
  
if (fechaProximaRenovacion != null && fechaVencimiento != null &&  
    !fechaProximaRenovacion.isAfter(LocalDate.now()) &&  
    !fechaVencimiento.isBefore(LocalDate.now())) {  
    boolean exito = enviarRecordatorio(codigo, requisitoLegal, comoSeAplica, porQueLoAplica, fechaEmision, fechaVencimiento, fechaProximaRenovacion, lblMensaje);  
    if (!exito) huboError = true;  
    enviados++;  
}  
}  
  
if (!huboError) {  
    resultado.append("Recordatorios enviados correctamente (" + enviados + " documentos para renovar).");  
} else {  
    resultado.append(str:"Algunos documentos fueron ignorados por formato de fecha inválido.");  
}
```

### 2. DIAGRAMA DE FLUJO (DF)





### 3. GRAFO DE FLUJO (GF)



#### 4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

##### RUTAS

**R1:** 1,2,3,4,6,7,8,2,9,10,11,13

**R2:** 1,2,3,4,6,7,8,2,9,10,12,13

**R3:** 1,2,3,2,9,10,12,13

**R4:** 1,2,3,4,2,9,10,12,13

**R5:** 1,2,3,4,5,2,9,10,12,13

#### 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predcados (decisiones)} + 1$   
 $V(G) = 6 + 1$   
 $V(G) = 7$
- $V(G) = A - N + 2$   
 $V(G) = 17 - 13 + 2$   
 $V(G) = 6$

DONDE:

**P:** Número de nodos predcado

**A:** Número de aristas

**N:** Número de nodos