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

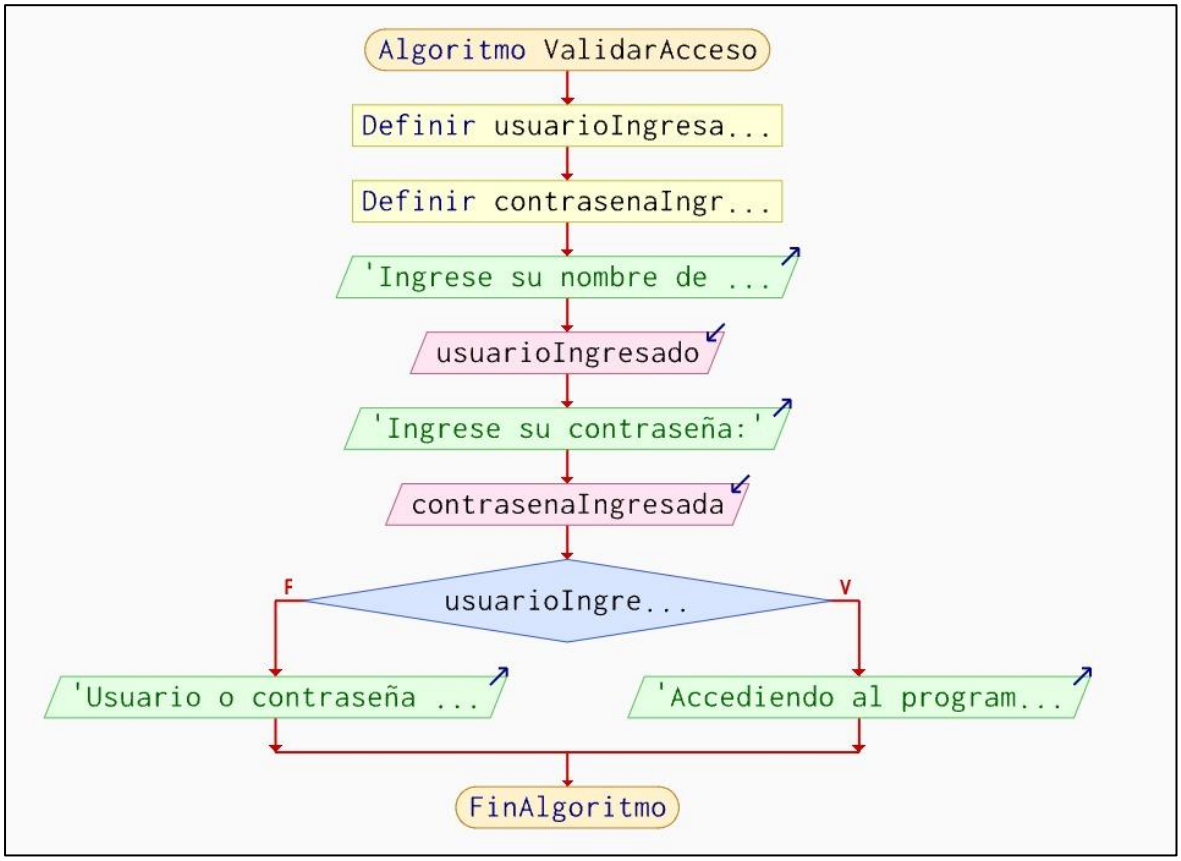
Fecha 2025-07-07

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

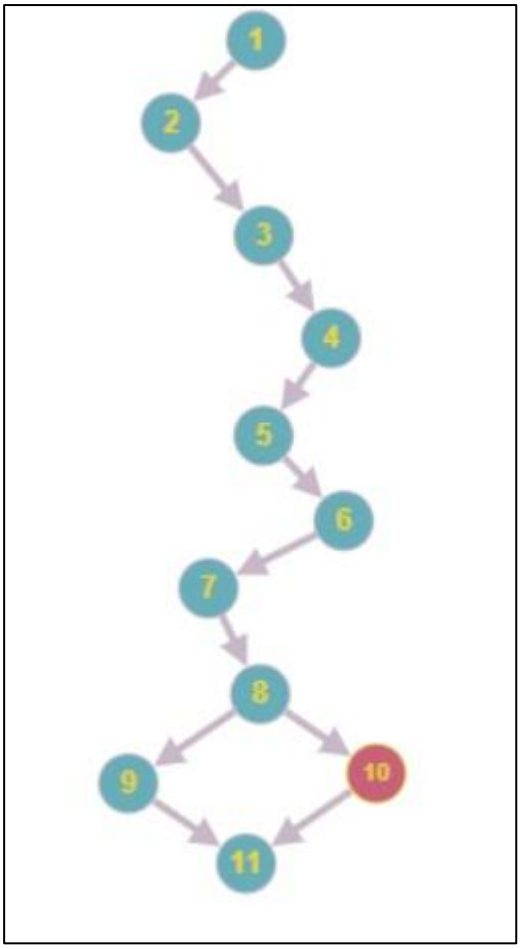
1. **CÓDIGO FUENTE**

```
private void verificarLogin(TextField txtUsuario, PasswordField txtContrasena, Label lblMensaje, Stage stage) {
    String usuario = txtUsuario.getText();
    String contrasena = txtContrasena.getText();
    if (usuarios.containsKey(usuario) && usuarios.get(usuario)[0].equals(contrasena)) {
        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. IDENTIFIACCIÒN 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) = 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

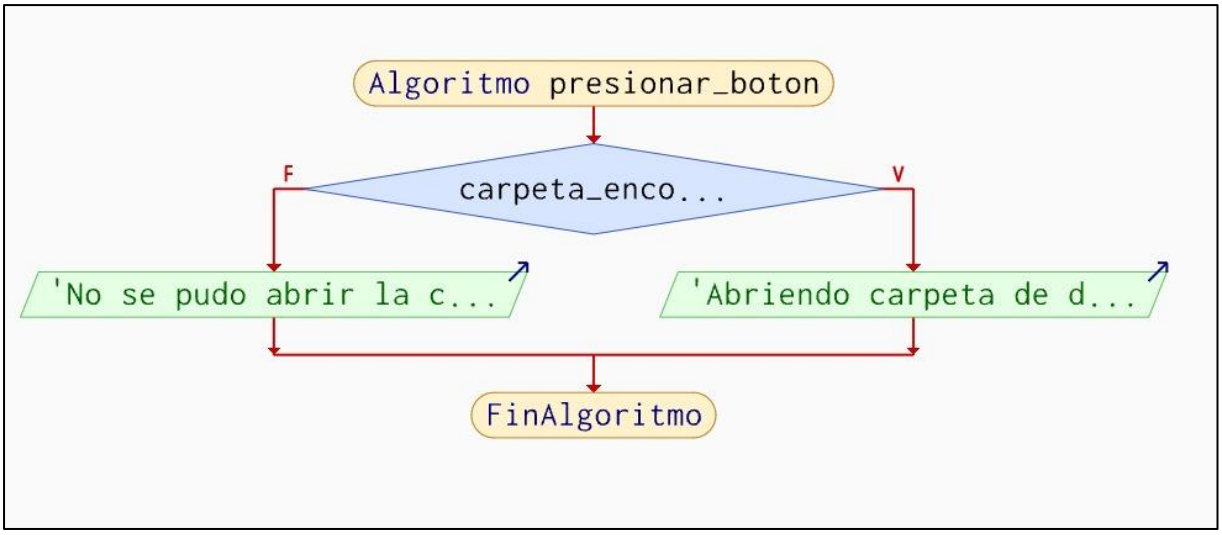
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

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

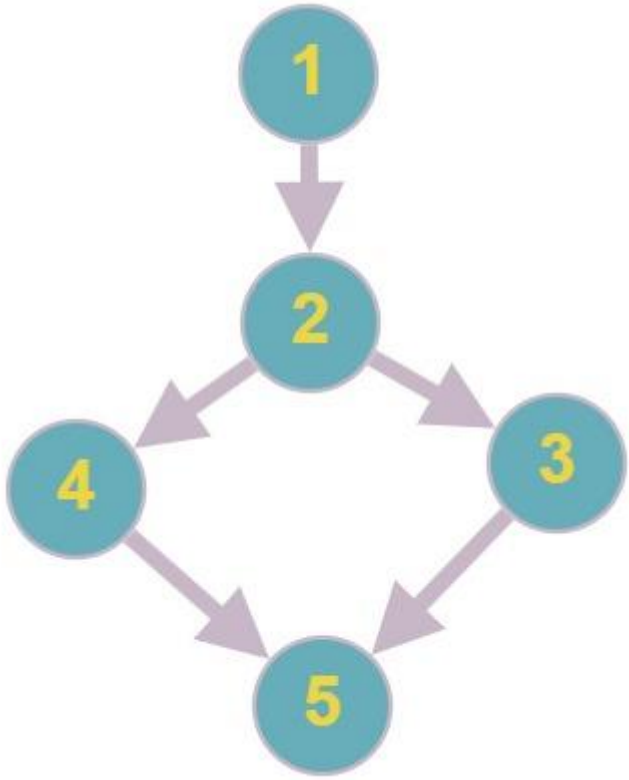
1. **CÓDIGO FUENTE**

```
btnFirmaContrato.setOnAction(_ -> {
    try {
        String folderPath = "C:\\Users\\santy\\OneDrive\\Documentos\\Carpetas 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. IDENTIFIACCIÒN 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) = número de nodos predicados (decisiones)+1
  V(G)= 1+1
  V(G)= 2

- V(G) = A – N + 2
  V(G)= 5 – 5 + 2
  V(G)= 2

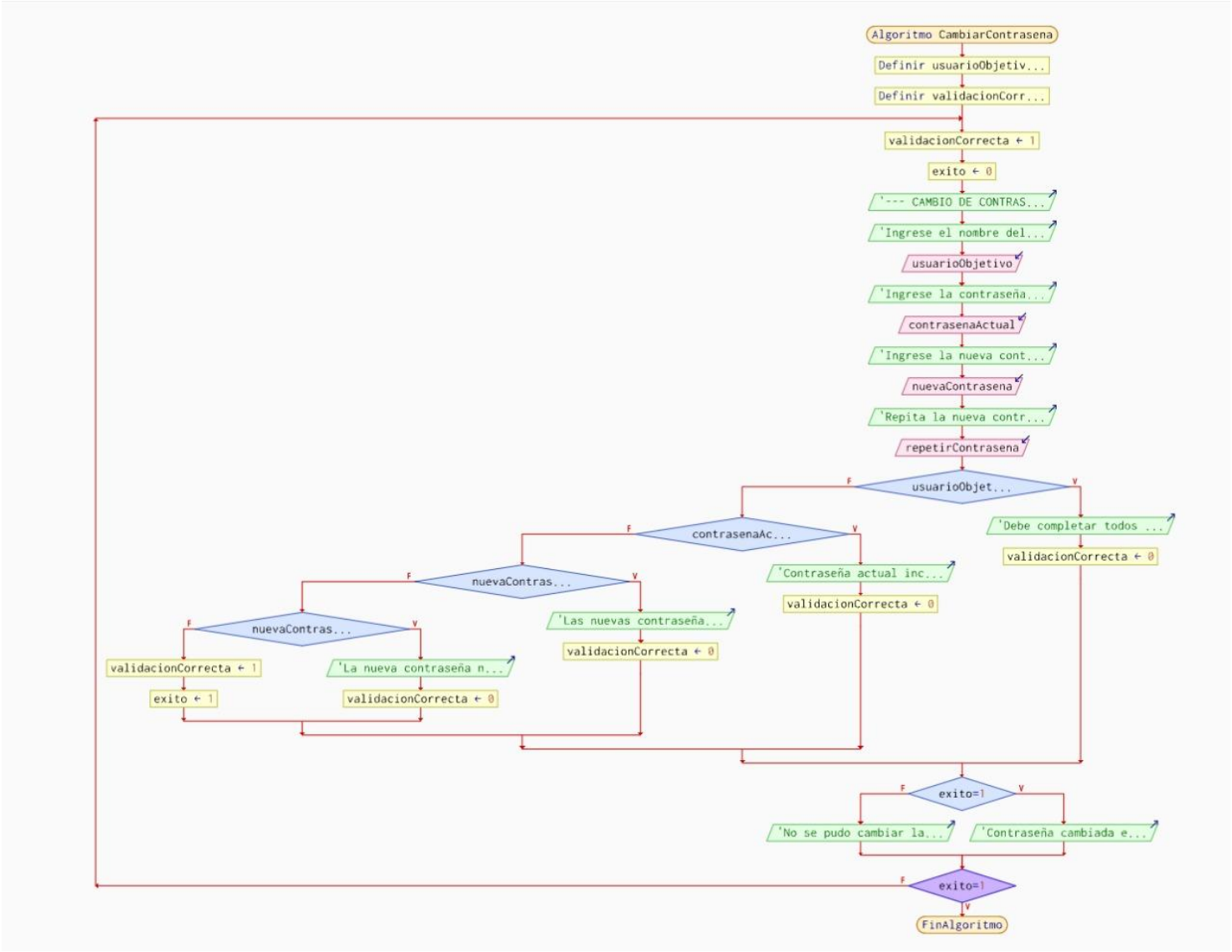
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de: Modificar las credenciales de los usuarios

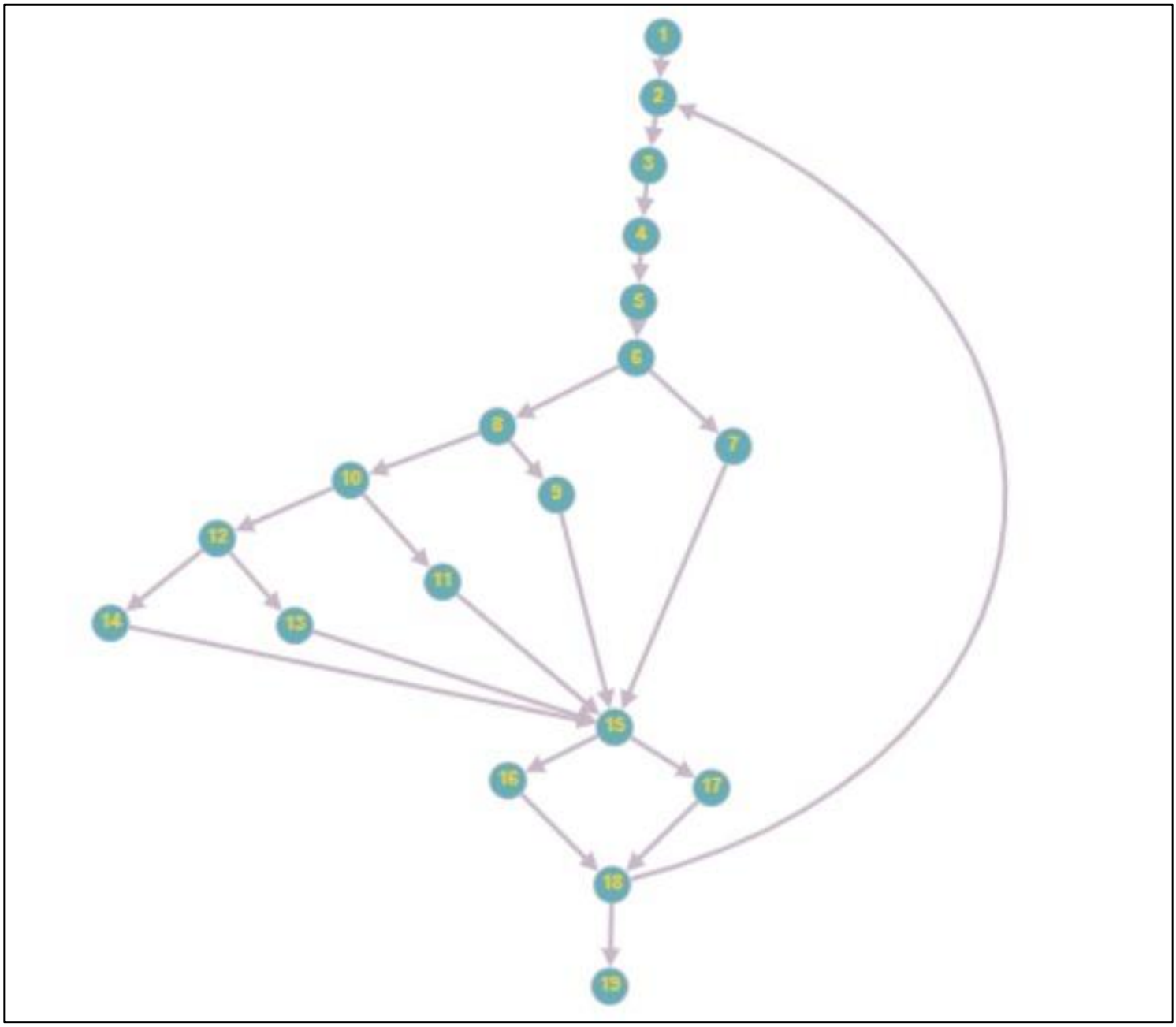
1. CÓDIGO FUENTE

```
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 || nuevaContrasen
    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. IDENTIFIACCIÒN 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) = número de nodos predicados (decisiones)+1
  V(G)= 5+1
  V(G)= 6

- V(G) = A – N + 2
  V(G)= 24 - 19 + 2
  V(G)= 3

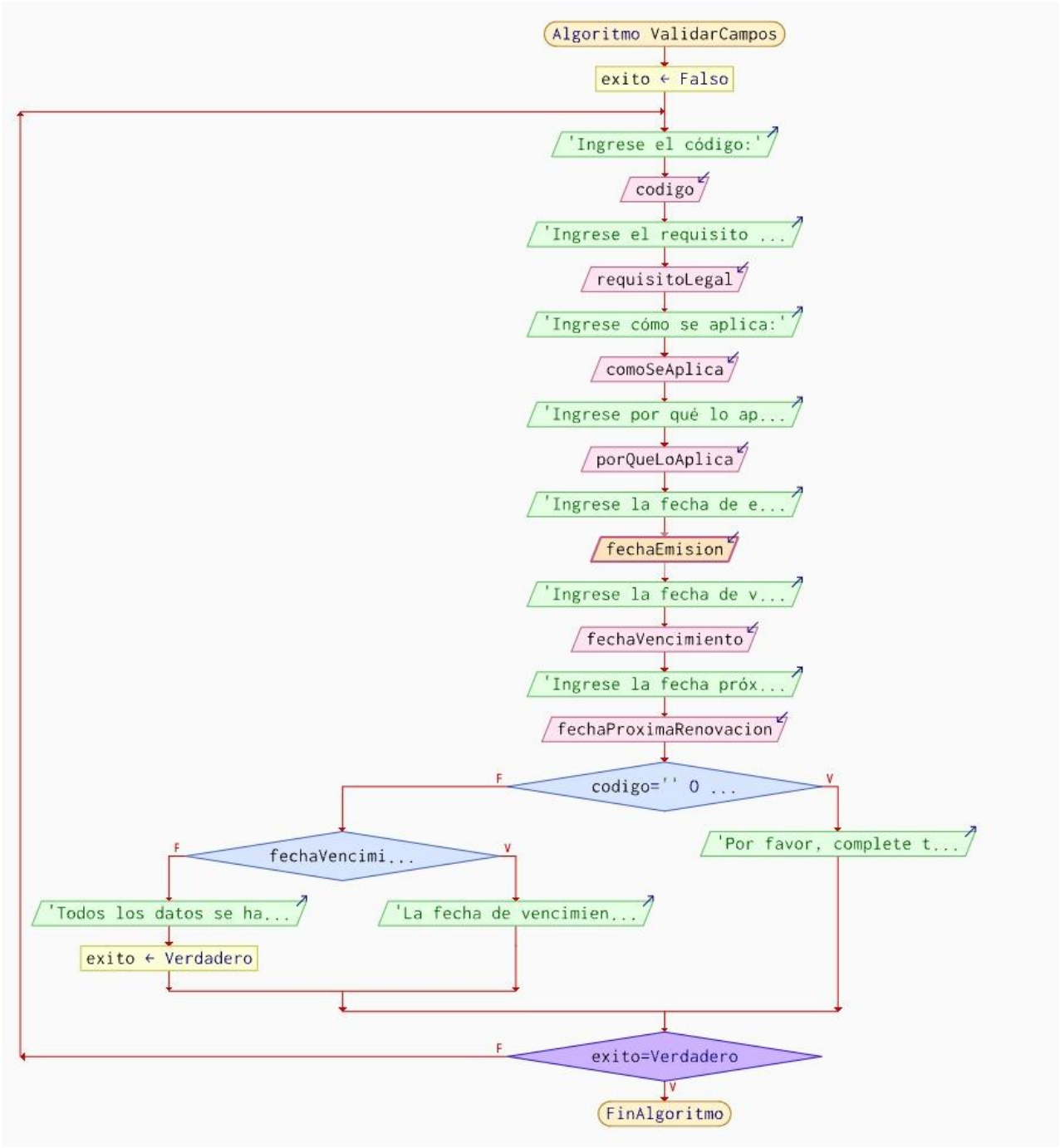
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

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

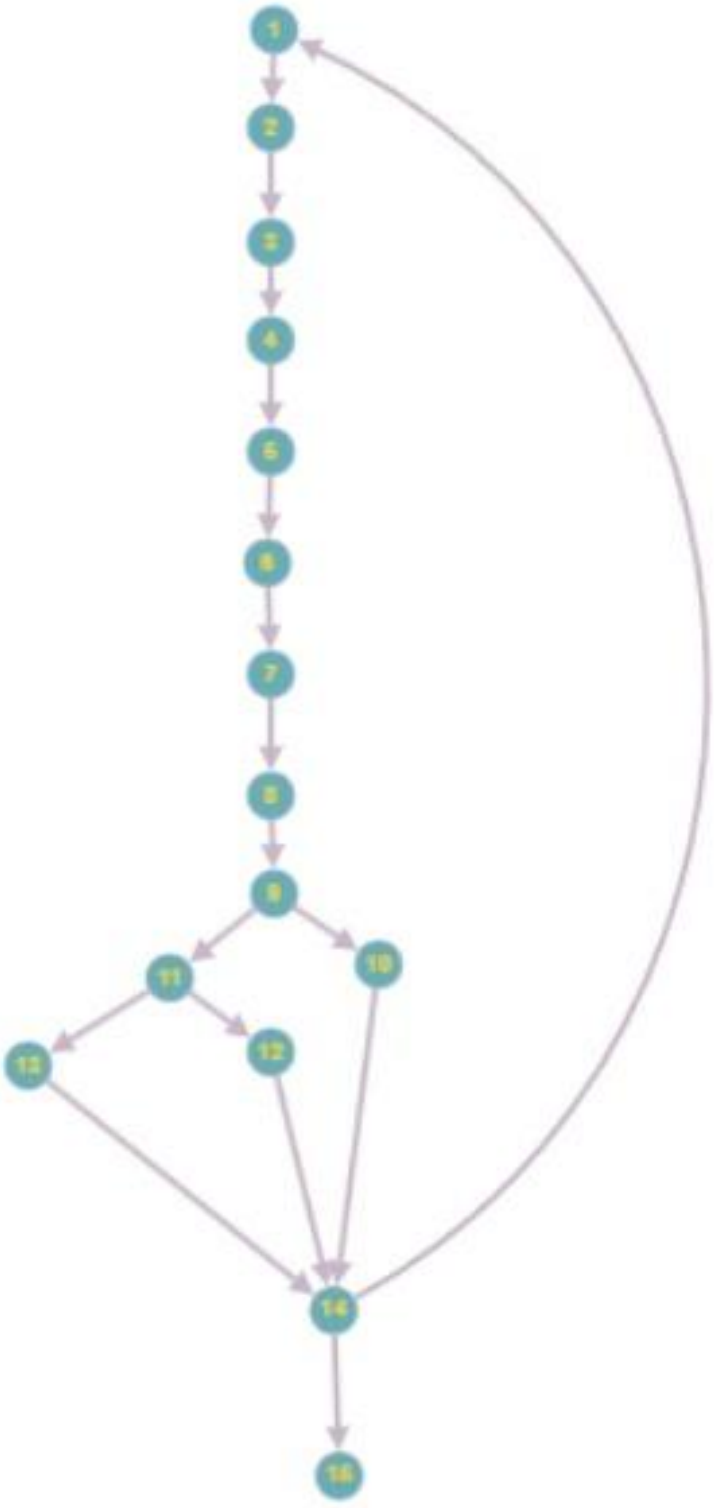
1. CÓDIGO FUENTE

```
if (codigo.isEmpty() || requisitoLegal.isEmpty() || comoSeAplica.isEmpty() ||
    porQueLoAplica.isEmpty() || fechaEmision == null || fechaVencimiento == null ||
    fechaProximaRenovacion == null) {
    DialogUtils.mostrarAviso(grid, mensaje:"Por favor, complete todos los campos.", esExito:false, onSuccess:null);
    return;
}

if (fechaVencimiento.isBefore(fechaEmision) || fechaProximaRenovacion.isBefore(fechaEmision)||fechaProximaRenovacion.isBefore(fechaVencimiento)) {
    DialogUtils.mostrarAviso(grid, mensaje:"La fecha de vencimiento y la fecha próxima de renovación no pueden ser anteriores a la fecha de emisión.", esExito:false, onSu
    return;
}

try {
    // Verificar si el código ya existe
    if (documentoService.existeDocumento(codigo)) {
        DialogUtils.mostrarAviso(grid, mensaje:"Ya existe un documento con este código.", esExito:false, onSuccess:null);
        return;
    }

    // Guardar documento usando el servicio
    boolean guardado = documentoService.guardarDocumento(
        codigo, requisitoLegal, comoSeAplica, porQueLoAplica,
        fechaEmision, fechaVencimiento, fechaProximaRenovacion,
        archivosSeleccionados
    );
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1,2,3,4,5,6,7,8,9,11,12,14,16
R2: No llegan al final

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados (decisiones)+1
  V(G)= 3+1
  V(G)= 4

- V(G) = A – N + 2
  V(G)= 17 - 16 + 2
  V(G)= 3

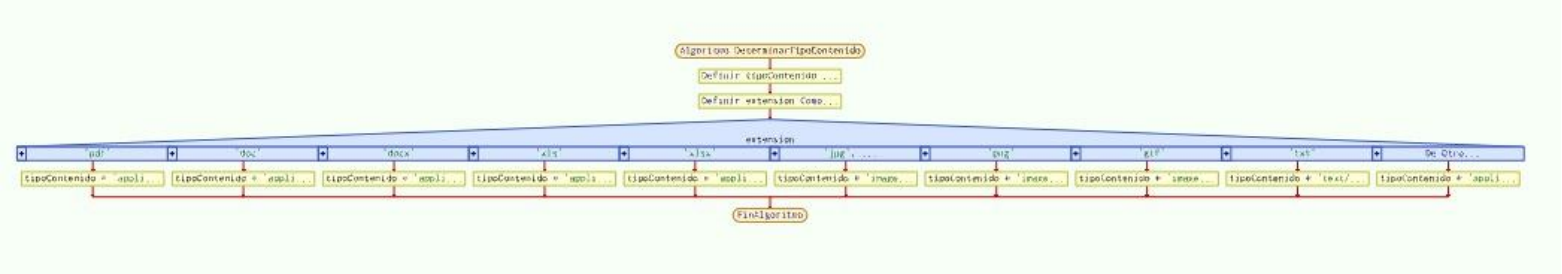
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de: Guardar la Información en el sistema

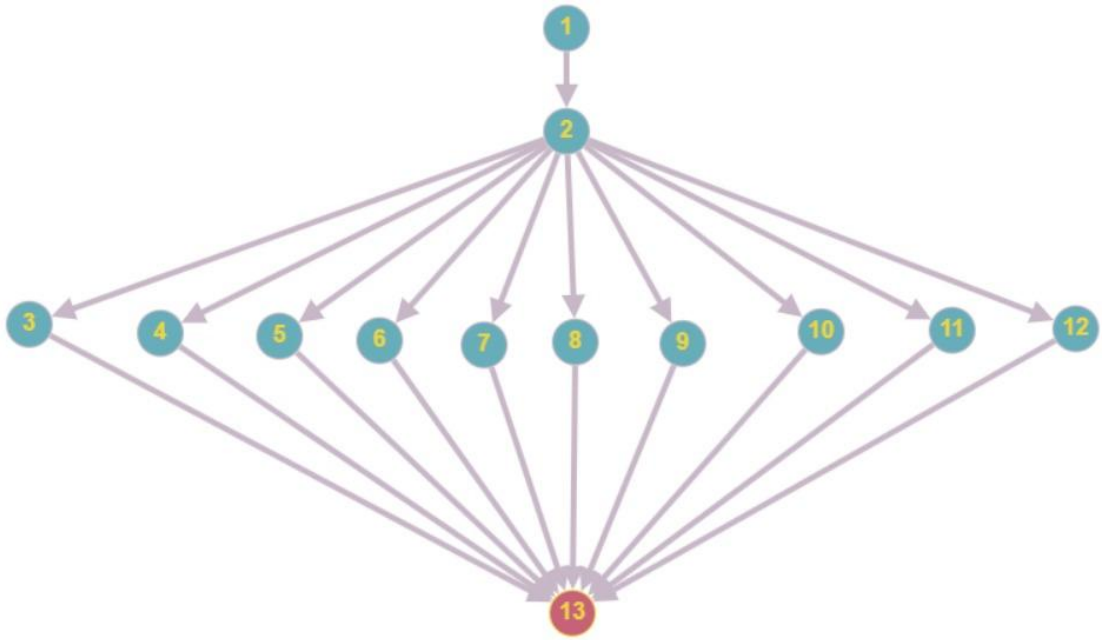
1. **CÓDIGO FUENTE**

```java
private String determinarTipoContenido(String nombreArchivo) {
    String extension = nombreArchivo.substring(nombreArchivo.lastIndexOf(ch:'.') + 1).toLowerCase();
    switch (extension) {
        case "pdf": return "application/pdf";
        case "doc": return "application/msword";
        case "docx": return "application/vnd.openxmlformats-officedocument.wordprocessingml.document";
        case "xls": return "application/vnd.ms-excel";
        case "xlsx": return "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
        case "jpg": case "jpeg": return "image/jpeg";
        case "png": return "image/png";
        case "gif": return "image/gif";
        case "txt": return "text/plain";
        default: return "application/octet-stream";
    }
```

2. **DIAGRAMA DE FLUJO (DF)**



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1,2,3,13
R2: 1,2,4,13
R3: 1,2,5,13
R4: 1,2,6,13
R5: 1,2,7,13
R6: 1,2,8,13
R7: 1,2,9,13
R8: 1,2,10,13
R9: 1,2,11,13
R10: 1,2,12,13

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

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

- V(G) = A – N + 2
  V(G)= 21 - 13 + 2
  V(G)= 10

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