Prueba de Caja Blanca

"Título de proyecto: Sistema de Inventario Medicina"

Integrantes:

Cesar Herrera

Kelly Montalvo

Matias Intriago

Fecha: 2025/12/03

Prueba caja blanca

RF N1

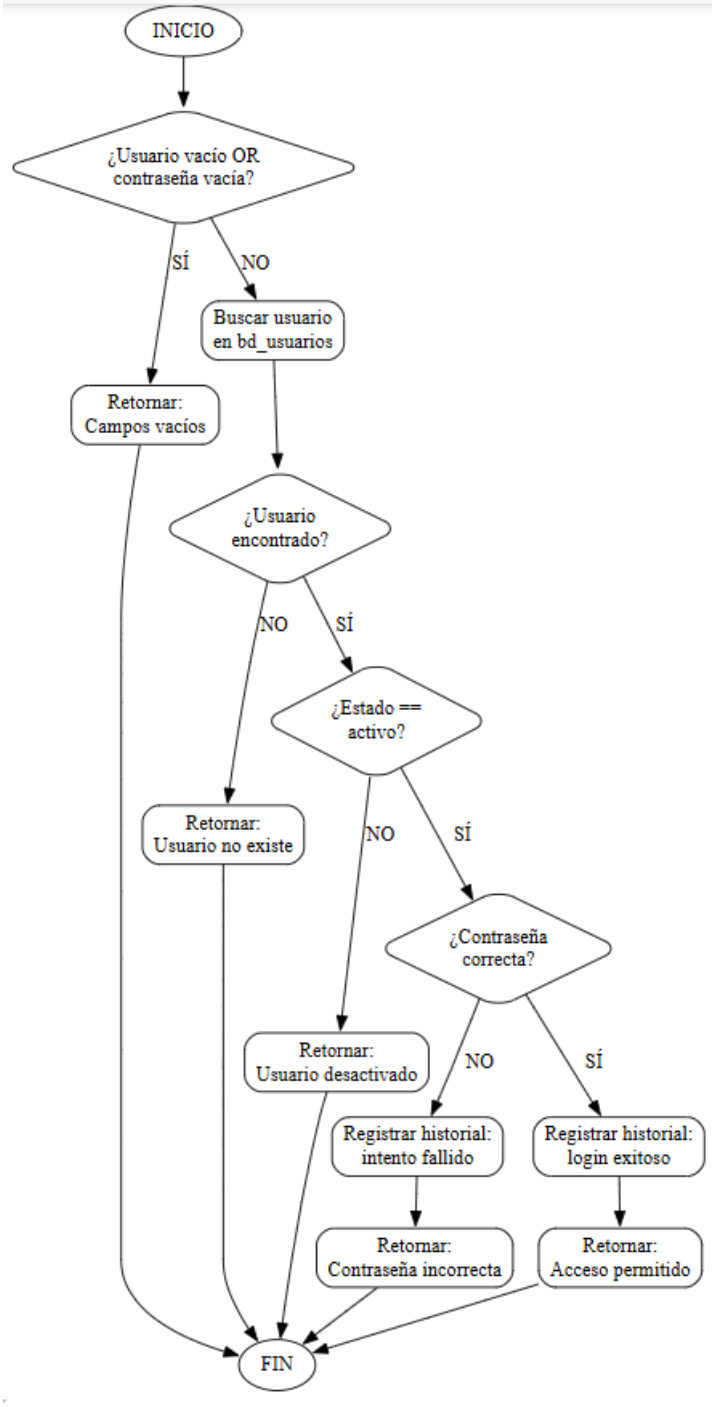
1. CÓDIGO FUENTE

Validación de verificación de datos de usuarios

```
public class SistemaAutenticacion {

    public Map<String, String> autenticarAdministrador(String usuario, String contrasena,
                                                        ArrayList<Administrador> bdUsuarios) {
        Map<String, String> resultado = new HashMap<>();


        if (usuario.equals("") || contrasena.equals("")) {
            resultado.put("status", "error");
            resultado.put("mensaje", "Campos vacíos");
            return resultado;
        }

        Administrador usuarioEncontrado = null;
        for (Administrador admin : bdUsuarios) {
            if (admin.getUsuario().equals(usuario)) {
                usuarioEncontrado = admin;
                break;
            }
        }

        if (usuarioEncontrado == null) {
            resultado.put("status", "error");
            resultado.put("mensaje", "Usuario no existe");
            return resultado;
        }

        if (!usuarioEncontrado.getEstado().equals("activo")) {
            resultado.put("status", "error");
            resultado.put("mensaje", "Usuario desactivado");
            return resultado;
```

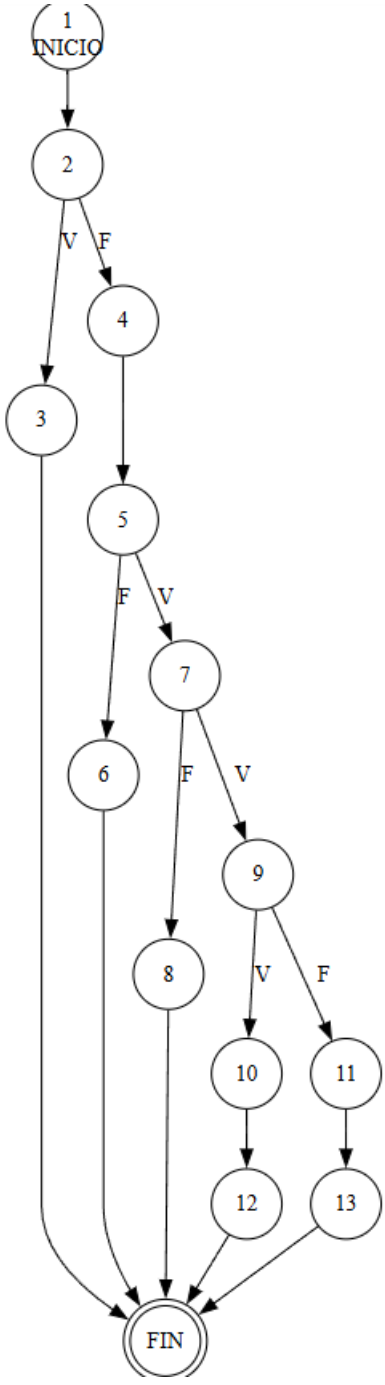
2. DIAGRAMA DE FLUJO (DF)

Diagrama de flujo del requisito 1



3. GRAFO DE FLUJO (GF)

Grafo de flujo del requisito 1



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 4

RUTAS

Ruta 1

Camino: 1 → 2 → 3 → FIN

Descripción: Alguno de los campos está vacío (usuario o contraseña).

Datos de prueba:

- usuario = ""
- contraseña = ""

Ruta 2

Camino: 1 → 2 → 4 → 5 → 6 → FIN

Descripción: El usuario ingresado no existe en la base de datos.

Datos de prueba:

- usuario = "admin_falso"
- contraseña = "123"

Ruta 3

Camino: 1 → 2 → 4 → 5 → 7 → 8 → FIN

Descripción: El usuario existe, pero su cuenta está desactivada o inactiva.

Datos de prueba:

- usuario = "admin1"
- estado = "inactivo"

Ruta 4

Camino: 1 → 2 → 4 → 5 → 7 → 9 → 11 → 13 → FIN

Descripción: El usuario está activo, pero la contraseña ingresada es incorrecta.

Datos de prueba:

- usuario = "admin1"
- contraseña = "incorrecta"

Ruta 5

Camino: 1 → 2 → 4 → 5 → 7 → 9 → 10 → 12 → FIN

Descripción: Autenticación exitosa (usuario válido, activo y contraseña correcta).

Datos de prueba:

- usuario = "admin1"
- contraseña = "correcta"

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

Nodos (N): Son todos los círculos numerados.

Nodos predicados (P): Son los nodos de decisión, que tienen más de una salida: -

A (aristas): Contando todas las flechas entre nodos.

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

- V(G) = A – N + 2

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

Datos del grafo de flujo

- **Nodos (N):** 13
- **Nodos predicados (P):** 4
  - Nodo 2: ¿Campos vacíos?
  - Nodo 5: ¿Usuario encontrado?
  - Nodo 7: ¿Usuario activo?
  - Nodo 9: ¿Contraseña correcta?
- **Aristas (A):** 16 (valor corregido)

Cálculo de la Complejidad Ciclomática
Método 1: V(G) = P + 1

Se suman los nodos de decisión más 1.

V(G) = 4 + 1 = 5

Método 2: V(G) = A - N + 2

Usa aristas, nodos y una constante.

V(G) = 16 - 13 + 2 = 5

Resultado final

Ambos métodos coinciden:

La complejidad ciclomática del grafo es: 5

Prueba de Caja Blanca

"Título de proyecto: Sistema de Inventario Medicina"

Integrantes:

César Herrera

Kelly Montalvo

Matias Intriago

Fecha: 2025/12/03

Prueba caja blanca
RF N2

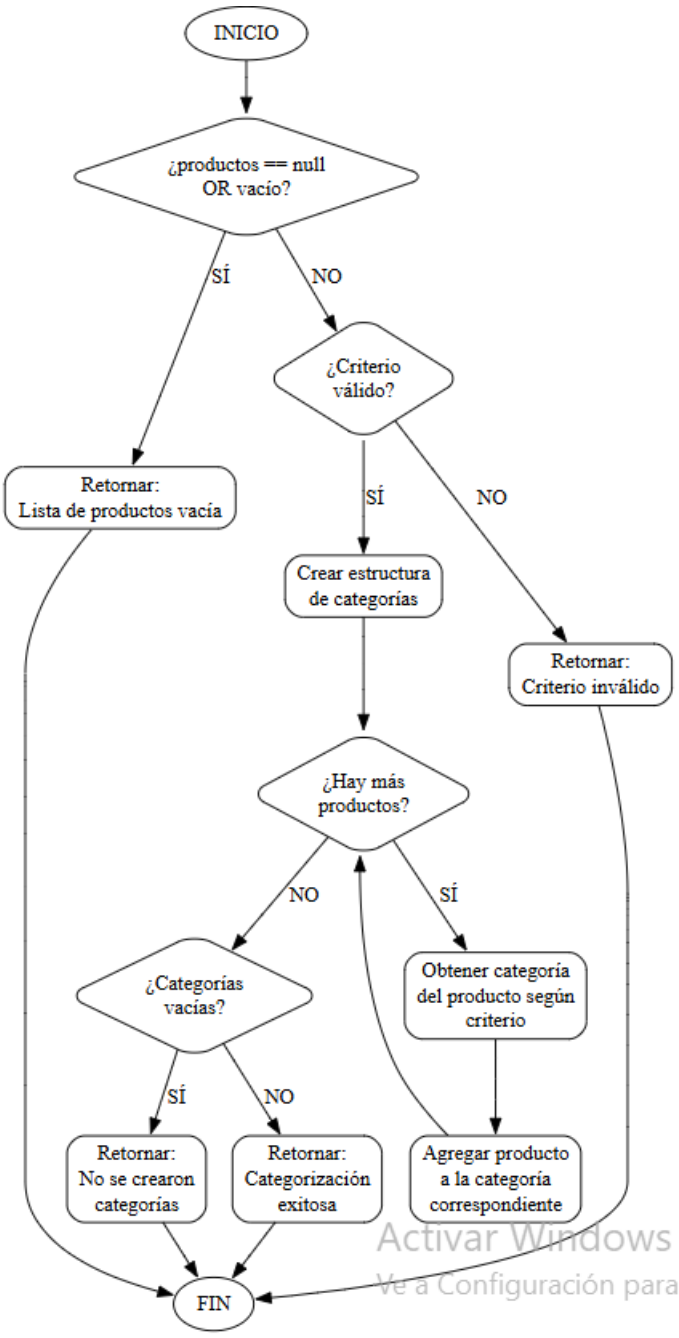
1. **CÓDIGO FUENTE**

Validación de categorización

```java
public class SistemaCategorizacion {

    private String codigo;
    private String nombre;
    private CategoriaMedicamento categoria;
    private int stock;
    private double precioUnitario;

    public SistemaCategorizacion(String codigo, String nombre,
                                 CategoriaMedicamento categoria,
                                 int stock, double precioUnitario) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.categoria = categoria;
        this.stock = stock;
        this.precioUnitario = precioUnitario;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public CategoriaMedicamento getCategoria() {
        return categoria;
    }

    public int getStock() {
        return stock;
    }

    public double getPrecioUnitario() {
        return precioUnitario;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }

    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }

    @Override
    public String toString() {
        return "Código: " + codigo +
               " | Nombre: " + nombre +
               " | Categoría: " + categoria +
               " | Stock: " + stock +
               " | Precio: " + precioUnitario;
    }
}
```

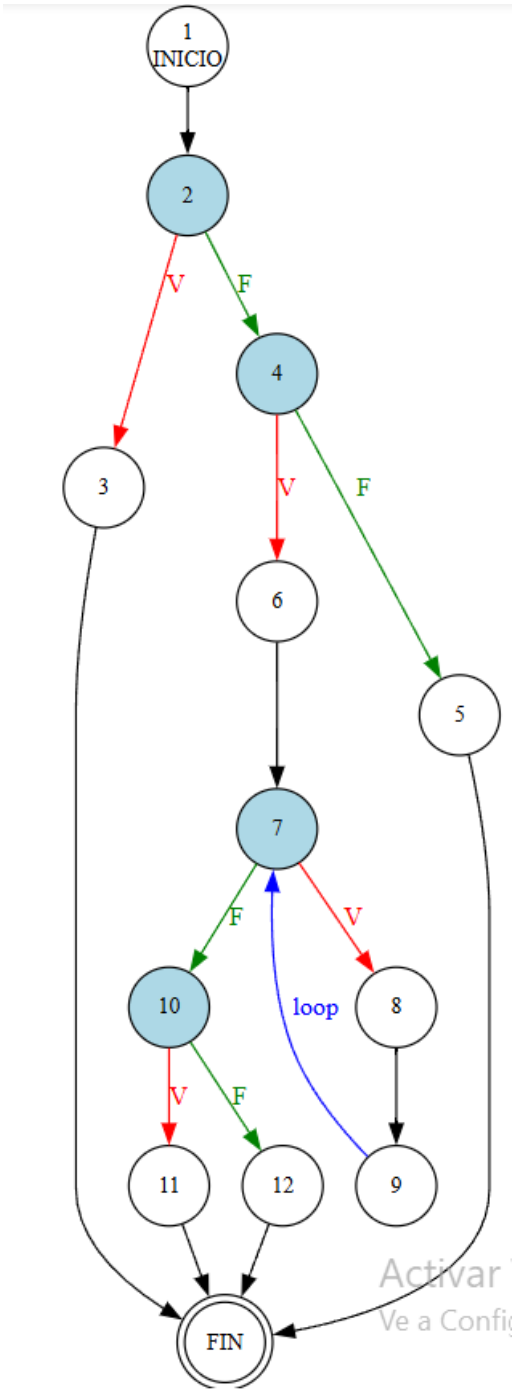
2. DIAGRAMA DE FLUJO (DF)

Diagrama de flujo del requisito 2



3. GRAFO DE FLUJO (GF)

Grafo de flujo del requisito 2



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 4
RUTAS

Ruta 1
Camino: 1 → 2 → 3 → FIN
Descripción: Lista de productos vacía.
Datos de prueba:
productos = null
productos.isEmpty() = true

Ruta 2
Camino: 1 → 2 → 4 → 5 → FIN
Descripción: Criterio de categorización inválido.
Datos de prueba:
criterio = "color" (no soportado)

Ruta 3
Camino: 1 → 2 → 4 → 6 → 7 → 10 → 11 → FIN
Descripción: No se crean categorías (la lista se procesa, pero no genera ninguna categoría).
Datos de prueba:
productos válidos, pero no producen categorías

Ruta 4
Camino: 1 → 2 → 4 → 6 → 7 → 8 → 9 → 7 → 10 → 12 → FIN
Descripción: Categorización exitosa (procesa productos y crea categorías).
Datos de prueba:
productos válidos
criterio = "categoria"

Ruta 5
Camino: 1 → 2 → 4 → 6 → 7 → 10 → 12 → FIN
Descripción: Hay estructura creada, pero casi no hay productos para procesar (solo uno).
Datos de prueba:
productos = [1 elemento]
categorización exitosa

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

Nodos (N): Son todos los círculos numerados.

Nodos predicados (P): Son los nodos de decisión, que tienen más de una salida: -

A (aristas): Contando todas las flechas entre nodos.

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

- V(G) = A – N + 2

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

Conteo de elementos:
Nodos (N): 12 nodos
Nodos predicados (P): 4 nodos de decisión
Nodo 2: ¿Lista vacía?
Nodo 4: ¿Criterio válido?
Nodo 7: ¿Hay más productos? (loop)
Nodo 10: ¿Categorías vacías?
Aristas (A): 16 aristas

Cálculo:
Método 1: V(G) = P + 1
V(G) = 4 + 1 = 5
Método 2: V(G) = A - N + 2
V(G) = 16 - 12 + 2 = 6
Recálculo con A = 15:
V(G) = 15 - 12 + 2 = 5
Resultado:
Complejidad Ciclomática = 5

Prueba de Caja Blanca

"Título de proyecto: Sistema de Inventario Medicina"

Integrantes:

Cesar Herrera

Kelly Montalvo

Matias Intriago

Fecha: 2025/12/03

Prueba caja blanca

RF N3

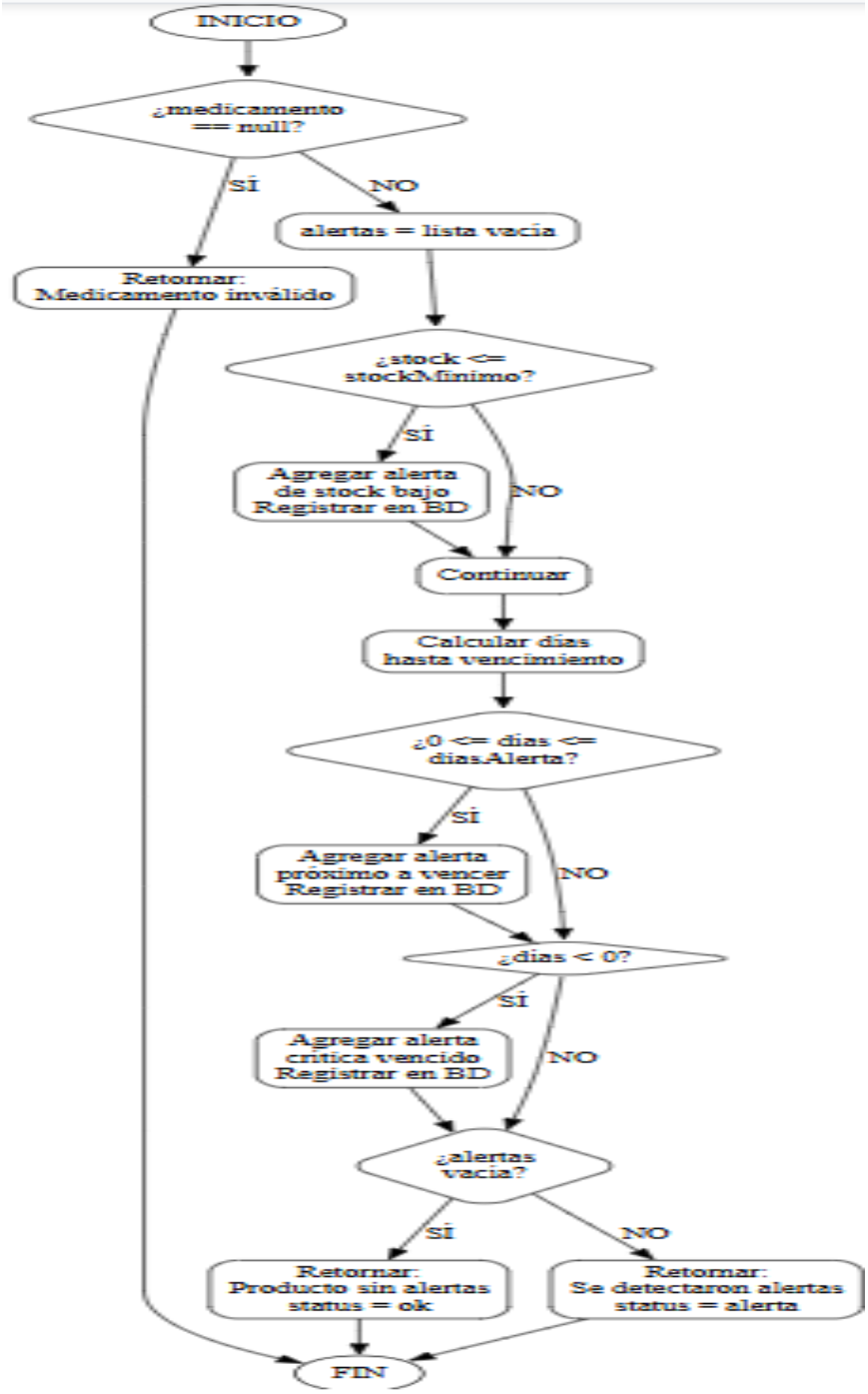
1. **CÓDIGO FUENTE**

Validación de stock y alarmas

```
public class SistemaCategorizacion {

    public static List<Alerta> verificar(Medicamento m, int diasAlerta) {

        if (m == null) {
            System.out.println("Medicamento inválido");
            return null;
        }

        List<Alerta> alertas = new ArrayList<>();

        if (m.stock <= m.stockMin)
            alertas.add(new Alerta("Stock bajo"));

        long dias = ChronoUnit.DAYS.between(LocalDate.now(), m.caducidad);

        if (dias >= 0 && dias <= diasAlerta)
            alertas.add(new Alerta("Próximo a vencer"));

        if (dias < 0)
            alertas.add(new Alerta("Producto vencido"));

        if (alertas.isEmpty()) {
            System.out.println("Producto sin alertas | status = ok");
        } else {
            System.out.println("Alertas detectadas | status = alerta");
        }

        return alertas;
    }

    public static void main(String[] args) {

        Medicamento m = new Medicamento(
                "Ibuprofeno", 5, 10,
                LocalDate.now().plusDays(3)
        );

        verificar(m, 7);
    }
}
```

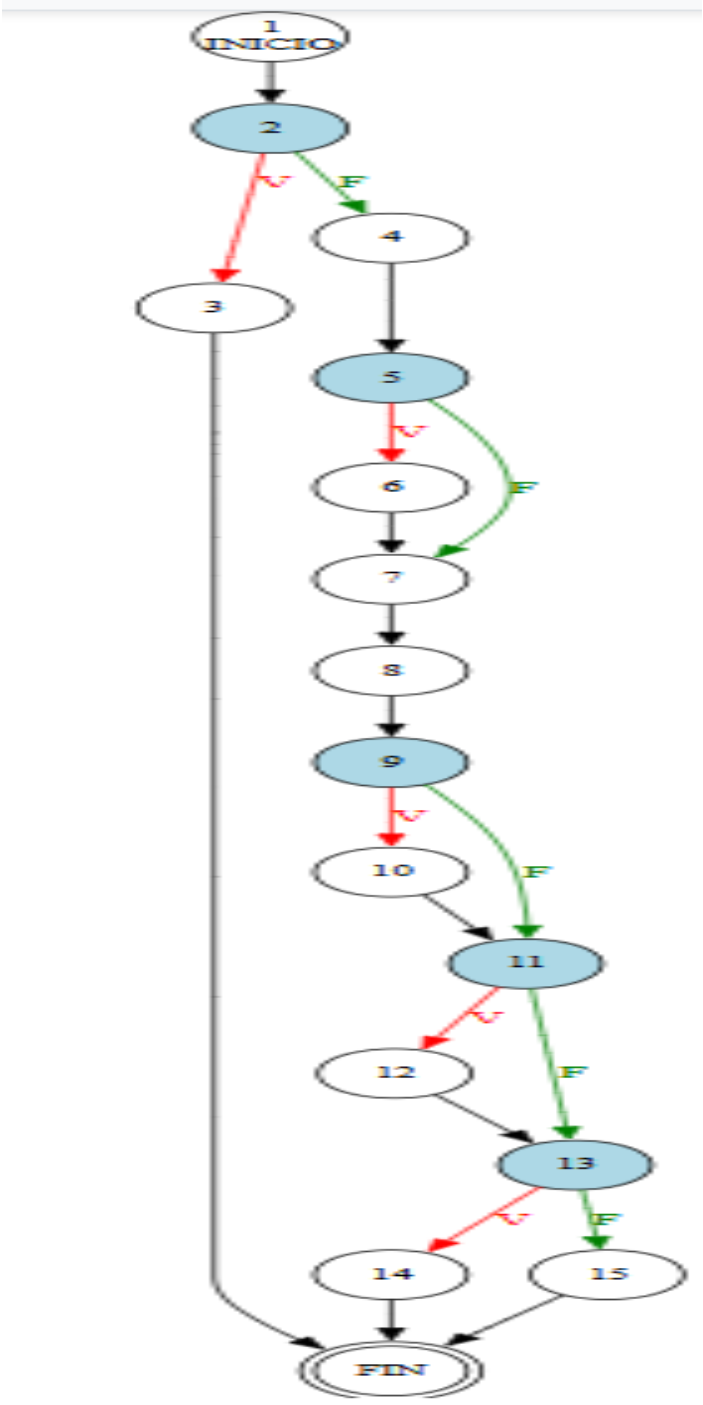
2. DIAGRAMA DE FLUJO (DF)

Diagrama de flujo del requisito 3



3. GRAFO DE FLUJO (GF)

Grafo de flujo del requisito 3



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 4
RUTAS
Ruta 1
Camino: 1 → 2 → 3 → FIN
Descripción: Medicamento nulo.
Datos de prueba:
medicamento = null

Ruta 2
Camino: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 13 → 14 → FIN
Descripción: Stock bajo, medicamento próximo a vencer y además vencido (todas las alertas activadas).
Datos de prueba:
stock = 5
stockMin = 10
días = 15
vencido = true

Ruta 3
Camino: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 11 → 13 → 14 → FIN
Descripción: Sin stock bajo, sin alerta de próximo vencimiento, sin alerta de vencimiento.
Datos de prueba:
stock = 50
stockMin = 10
días = 60

Ruta 4
Camino: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11 → 13 → 15 → FIN
Descripción: Solo alerta de stock bajo.
Datos de prueba:
stock = 5
stockMin = 10
días = 60

Ruta 5
Camino: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 13 → 15 → FIN
Descripción: Solo alerta de próximo vencimiento.
Datos de prueba:
stock = 50
stockMin = 10
días = 20

Ruta 6
Camino: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 13 → 15 → FIN
Descripción: Solo alerta de vencimiento.
Datos de prueba:
stock = 50
stockMin = 10
días = −10

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

Nodos (N): Son todos los círculos numerados.

Nodos predicados (P): Son los nodos de decisión, que tienen más de una salida: -

A (aristas): Contando todas las flechas entre nodos.

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

V/G) = P + 1

- V(G) = A – N + 2

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

Datos del grafo de flujo
Nodos (N): 15
Nodos predicados (P): 5
Nodo 2: ¿medicamento == null?
Nodo 5: ¿stock ≤ stockMínimo?
Nodo 9: ¿0 ≤ días ≤ diasAlerta?
Nodo 11: ¿días < 0?
Nodo 13: ¿alertas vacía?
Aristas (A): Se identificaron inicialmente 18, pero el grafo real contiene 19.

Cálculo de la Complejidad Ciclomática
Método 1: V(G) = P + 1
Suma de nodos de decisión más 1:
V(G) = 5 + 1 = 6

Método 2: V(G) = A - N + 2
Usando aristas y nodos:
V(G) = 18 - 15 + 2 = 5
Como hay una discrepancia, se ajusta el número correcto de aristas a 19:
V(G) = 19 - 15 + 2 = 6

Resultado final
La complejidad ciclomática del grafo es: 6