

Prueba de Caja Blanca

“Generación de proformas MarcallTex”

Integrantes:

Cañola Kevin

Marcalla Cristhian

Lugmaña Mateo

Tasiguano Eduardo

CONTROL DE VERSIONAMIENTO DE PRUEBAS CB

Versión	Fecha	Responsable	Aprobado por
PCB_V1.0.0.docx			

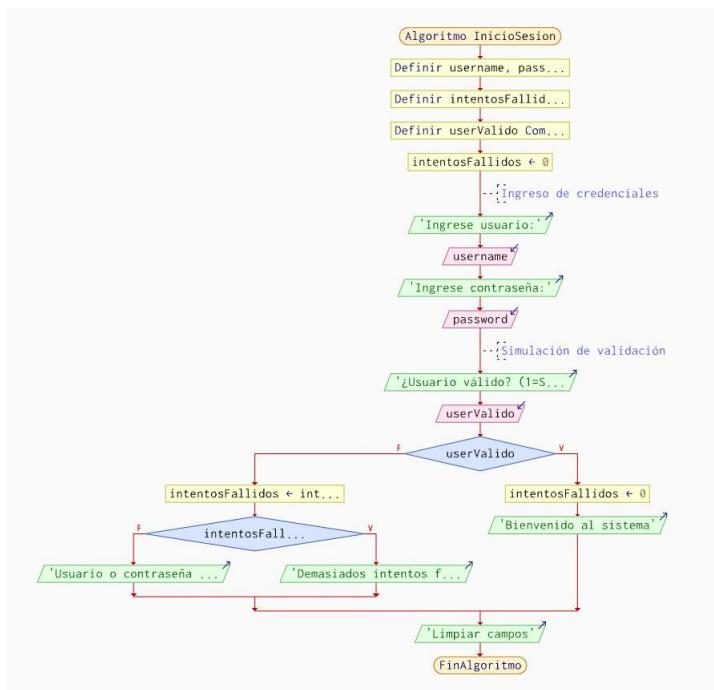
Prueba caja blanca

RF N1^a INICIO DE SESIÓN

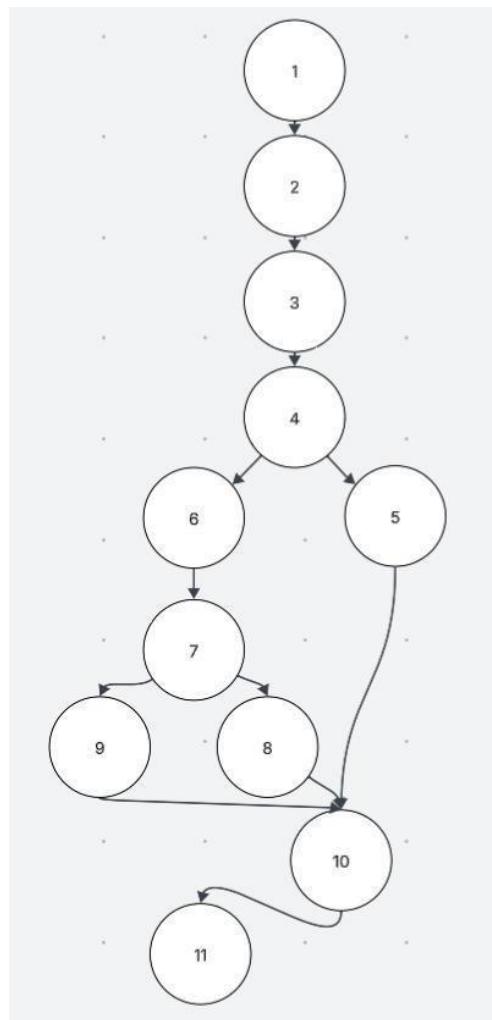
1. CÓDIGO FUENTE

```
35      Private Sub btnIngresar_Click(sender As Object, e As EventArgs) Handles btnIngresar.Click
36          Dim username As String = txtUsuario.Text.Trim()
37          Dim password As String = txtContraeña.Text.Trim()
38
39          Dim user = Usuario.ValidarLogin(username, password)
40
41          If user IsNot Nothing Then
42              intentosFallidos = 0
43
44              MsgBox("Bienvenido " & user.usuario, MsgBoxStyle.Information, "BIENVENIDO AL SISTEMA")
45
46              Dim admin As New ApartadoAdministrador()
47
48              ' ✓ flujo correcto
49              Me.Hide()
50              admin.Show()
51
52          Else
53              intentosFallidos += 1
54
55              If intentosFallidos >= 3 Then
56                  MsgBox("Demasiados intentos fallidos.", MsgBoxStyle.Critical, "Error")
57                  btnIngresar.Enabled = False
58                  bloqueoTimer.Start()
59              Else
60                  MsgBox("Usuario o contraseña incorrectos.", MsgBoxStyle.Critical, "Error")
61              End If
62          End If
63
64          LimpiarCampos()
65      End Sub
66
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 4 → 5 → 10 → 11	Usuario correcto, acceso concedido
R2	1 → 2 → 3 → 4 → 6 → 7 → 9 → 10 → 11	Credenciales incorrectas, menos de 3 intentos
R3	1 → 2 → 3 → 4 → 6 → 7 → 8 → 10 → 11	Credenciales incorrectas, tercer intento → bloqueo

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática del Grafo de Flujo se calcula para determinar el número de caminos independientes del sistema.

Datos del grafo

- **N (Número de nodos):** 11
- **P (Número de nodos predicados):** 2
- **A (Número de aristas):** 12

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 2 + 1 = 3$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 12 - 11 + 2 = 3$$

Resultado

La complejidad ciclomática del requisito **Inicio de Sesión** es **3**, lo que indica que existen **tres caminos básicos independientes**.

DONDE

P: Número de nodos predicado

A: Número de aristas

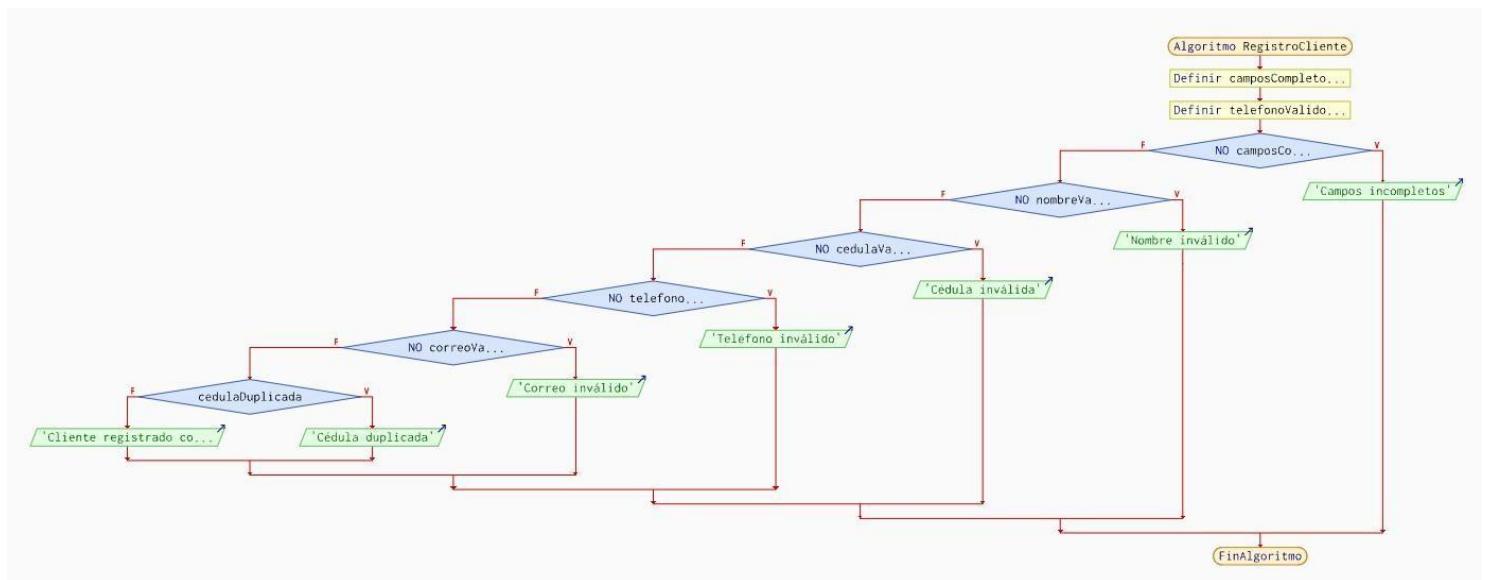
N: Número de nodos

RF N2^a REGISTRO DE CLIENTES

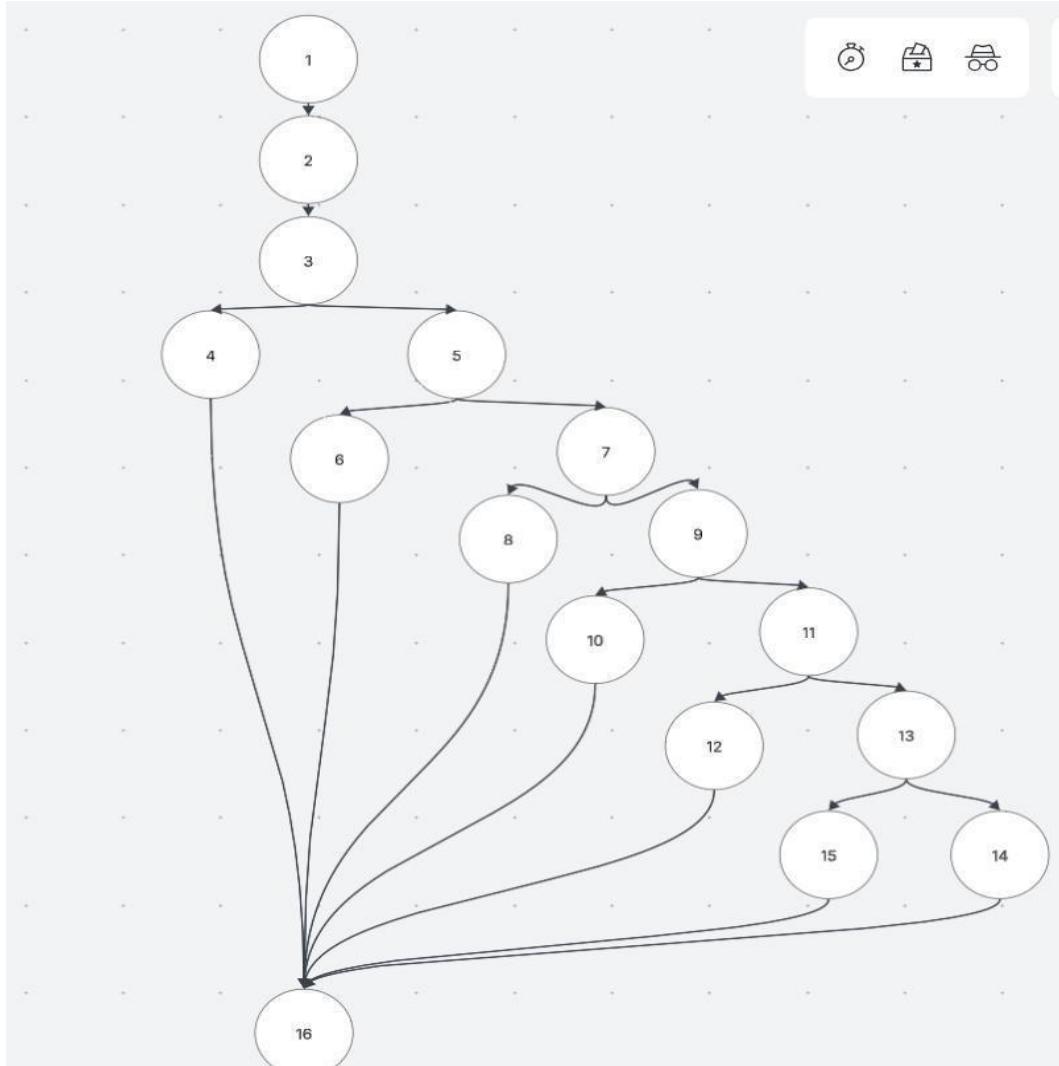
6. CÓDIGO FUENTE

```
1 Private Sub btnRegistrar_Click(sender As Object, e As EventArgs) Handles btnRegistrar.Click
2     Try
3         If txtNombre.Text.Trim() = "" OrElse
4             txtCedula.Text.Trim() = "" OrElse
5                 txtTelefono.Text.Trim() = "" OrElse
6                     txtCorreo.Text.Trim() = "" OrElse
7                         txtDireccion.Text.Trim() = "" Then
8                             MessageBox.Show("Por favor, complete todos los campos.")
9                             Exit Sub
10                        End If
11
12                        If Not Regex.IsMatch(txtNombre.Text.Trim(), "^[A-Za-zÁÉÍÓÚáéíóúññ ]+$") Then
13                            MessageBox.Show("El nombre solo puede contener letras.")
14                            Exit Sub
15                        End If
16
17                        If Not CedulaValida(txtCedula.Text.Trim()) Then
18                            MessageBox.Show("La cédula ingresada no es válida.")
19                            Exit Sub
20                        End If
21
22                        If Not Regex.IsMatch(txtTelefono.Text.Trim(), "^\d{10}$") Then
23                            MessageBox.Show("El teléfono debe contener 10 dígitos.")
24                            Exit Sub
25                        End If
26
27                        If Not Regex.IsMatch(txtCorreo.Text.Trim(), "^[a-zA-Z0-9._%+-]+@[gmail|outlook|hotmail|yahoo]\.(com|es)$") Then
28                            MessageBox.Show("Ingrese un correo válido.")
29                            Exit Sub
30                        End If
31
32                        Dim existente = colección.Find(filtroCedula).FirstOrDefault()
33                        If existente IsNot Nothing Then
34                            MessageBox.Show("Ya existe un cliente registrado con esta cédula.")
35                            Exit Sub
36                        End If
37
38                        colección.InsertOne(nuevoCliente)
39                        MessageBox.Show("Cliente registrado correctamente.")
40                        LimpiarCampos()
41
42                        Catch ex As Exception
43                            MessageBox.Show("Error al registrar cliente.")
44                        End Try
45
46 End Sub
```

7. DIAGRAMA DE FLUJO (DF)



8. GRAFO DE FLUJO (GF)



9. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 5 → 7 → 9 → 11 → 13 → 15 → 16	Registro exitoso del cliente
R2	1 → 2 → 3 → 4 → 16	Campos incompletos
R3	1 → 2 → 3 → 5 → 6 → 16	Nombre inválido
R4	1 → 2 → 3 → 5 → 7 → 8 → 16	Cédula inválida
R5	1 → 2 → 3 → 5 → 7 → 9 → 10 → 16	Teléfono inválido
R6	1 → 2 → 3 → 5 → 7 → 9 → 11 → 12 → 16	Correo inválido
R7	1 → 2 → 3 → 5 → 7 → 9 → 11 → 13 → 14 → 16	Cédula duplicada

10. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática del Grafo de Flujo se calcula para determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 16
- P (Número de nodos predicados): 6
- A (Número de aristas): 21

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 6 + 1 = 7$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 21 - 16 + 2 = 7$$

Resultado

La complejidad ciclomática del requisito **Registro de Clientes** es 7, lo que indica que existen **siete caminos básicos independientes**.

DONDE

P: Número de nodos predicado

A: Número de aristas

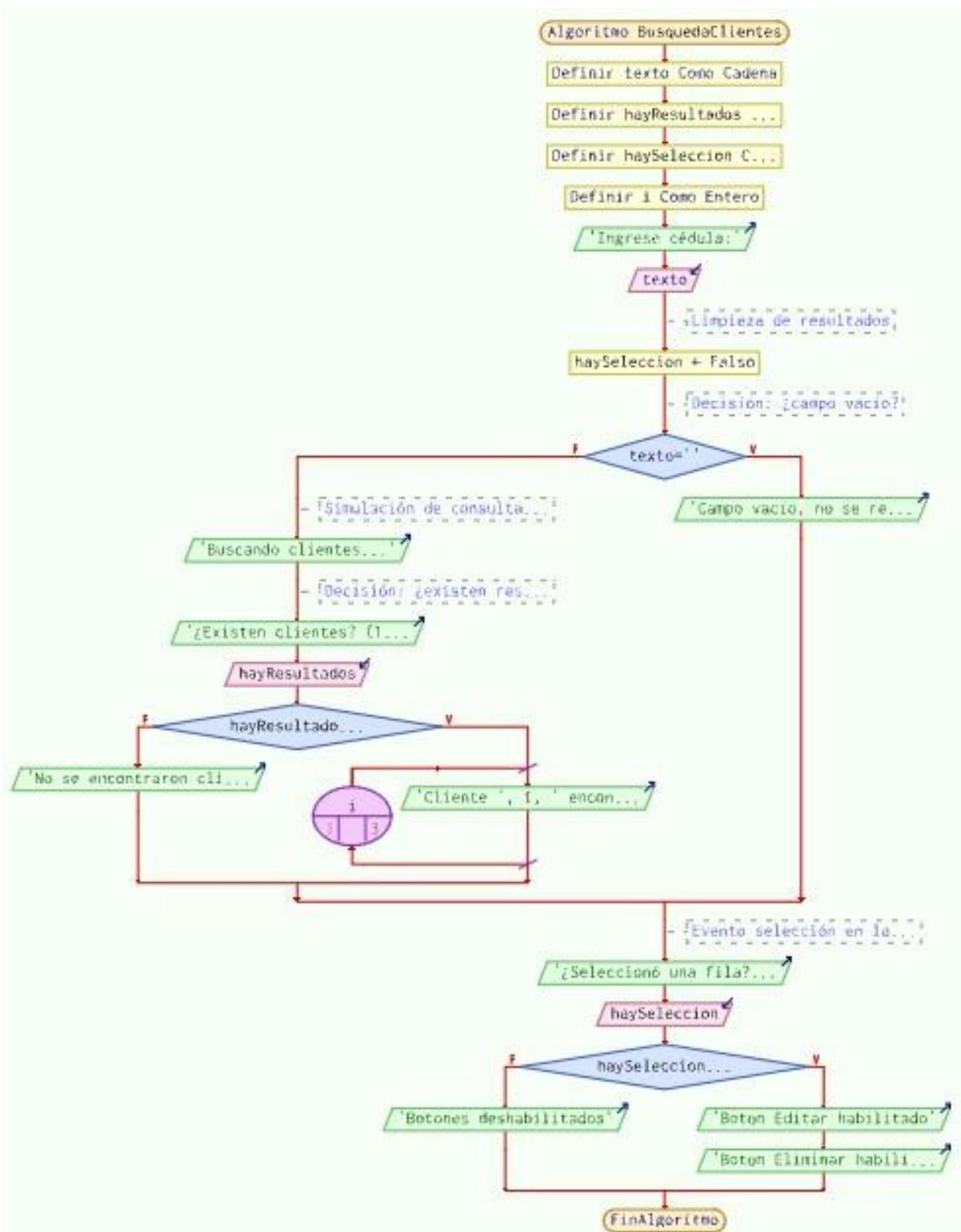
N: Número de nodos

RF N3ª BUSQUEDA CLIENTES

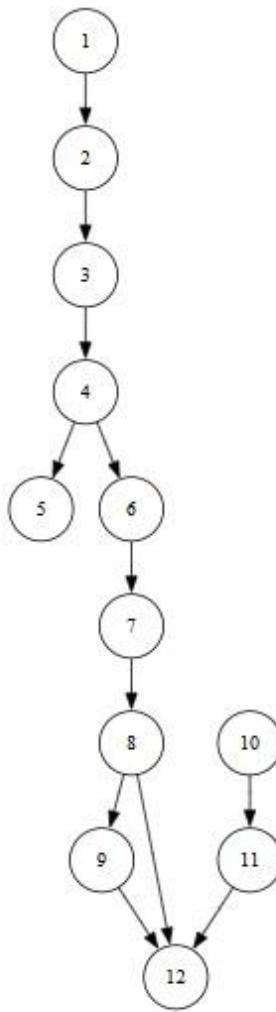
1. CÓDIGO FUENTE

```
39      ' ====== BUSCAR ======
40      3 referencias
41      Private Sub BuscarCliente()
42          Dim texto As String = txtCedula.Text.Trim()
43
44          dgvResultados.Rows.Clear()
45          btnEditar.Enabled = False
46          btnEliminar.Enabled = False
47
48          If texto = "" Then Exit Sub
49
50          Dim filtro = Builders(Of BsonDocument).Filter.Regex(
51              "cedula",
52              New BsonRegularExpression("^" & texto)
53          )
54
55          Dim clientes = clientesCollection.Find(filtro).ToList()
56
57          For Each cliente In clientes
58              dgvResultados.Rows.Add(
59                  cliente("cedula").ToString(),
60                  cliente("nombre").ToString(),
61                  cliente("telefono").ToString(),
62                  cliente("correo").ToString(),
63                  cliente("direccion").ToString()
64              )
65          Next
66      End Sub
67
68      ' ====== BUSCAR AUTOMÁTICO ======
69      0 referencias
70      Private Sub txtCedula_TextChanged(sender As Object, e As EventArgs) Handles txtCedula.TextChanged
71          BuscarCliente()
72      End Sub
73
74      ' ====== SOLO NÚMEROS ======
75      0 referencias
76      Private Sub txtCedula_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtCedula.KeyPress
77          If Not Char.IsControl(e.KeyChar) AndAlso Not Char.IsDigit(e.KeyChar) Then
78              e.Handled = True
79          End If
80      End Sub
81
82      ' ====== ☺ EVENTO CORRECTO ☺ ======
83      0 referencias
84      Private Sub dgvResultados_SelectionChanged(sender As Object, e As EventArgs) _
85          Handles dgvResultados.SelectionChanged
86
87          Dim haySeleccion As Boolean = dgvResultados.SelectedRows.Count > 0
88          btnEditar.Enabled = haySeleccion
89          btnEliminar.Enabled = haySeleccion
90      End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 4 → 5	Campo vacío, no se realiza búsqueda
R2	1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → 12	Texto ingresado con resultados encontrados
R3	1 → 2 → 3 → 4 → 6 → 7 → 8 → 12	Texto ingresado sin resultados
R4	10 → 11 → 12	Selección de fila y habilitación de botones

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 12
- P (Número de nodos predicados): 3
- A (Número de aristas): 14

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 3 + 1 = 4$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 14 - 12 + 2 = 4$$

Resultado

La complejidad ciclomática del requisito **Búsqueda de Clientes** es **4**, lo que indica que existen **cuatro caminos básicos independientes** que deben ser cubiertos por las pruebas de caja blanca.

DONDE

P: Número de nodos predicado

A: Número de aristas

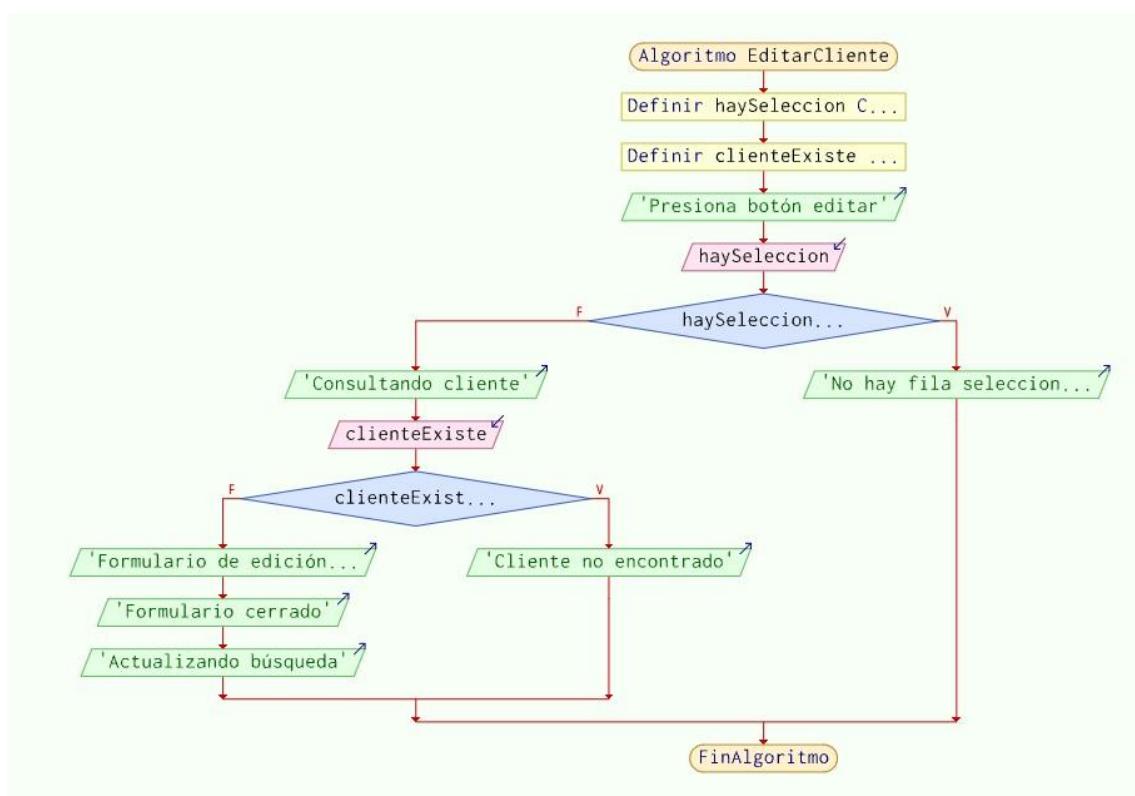
N: Número de nodos

RF N4ª MODIFICACION DE CLIENTES

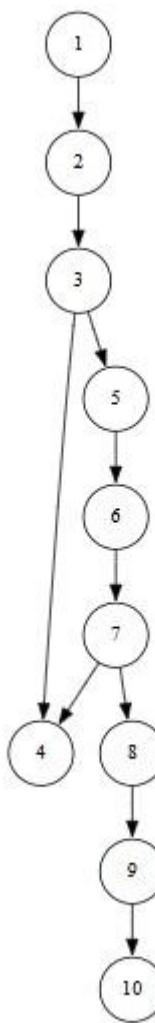
1. CÓDIGO FUENTE

```
88      ' ===== EDITAR =====
89      Private Sub btnEditar_Click(sender As Object, e As EventArgs) Handles btnEditar.Click
90          If dgvResultados.SelectedRows.Count = 0 Then Exit Sub
91
92          Dim fila = dgvResultados.SelectedRows(0)
93          Dim cedula As String = fila.Cells("cedula").Value.ToString()
94
95          Dim cliente = clientesCollection.Find(
96              New BsonDocument("cedula", cedula)
97          ).FirstOrDefault()
98
99          If cliente Is Nothing Then Exit Sub
100
101         Dim frm As New editarCliente(
102             cliente("cedula").ToString(),
103             cliente("nombre").ToString(),
104             cliente("telefono").ToString(),
105             cliente("correo").ToString(),
106             cliente("direccion").ToString(),
107             If(cliente.Contains("infoAdicional"), cliente("infoAdicional").ToString(), ""))
108
109
110         frm.ShowDialog()
111         BuscarCliente()
112     End Sub
113
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 4	No hay fila seleccionada
R2	1 → 2 → 3 → 5 → 6 → 7 → 4	Cliente no encontrado
R3	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 10	Modificación correcta del cliente

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- **N (Número de nodos):** 10
- **P (Número de nodos predicados):** 2

- A (Número de aristas): 11

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 2 + 1 = 3$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 11 - 10 + 2 = 3$$

Resultado

La complejidad ciclomática del requisito **Editar Cliente** es 3, lo que indica que existen **tres caminos independientes** que deben ser probados.

DONDE

P: Número de nodos predicado

A: Número de aristas

N: Número de nodos

RF N5ª ELIMINACION DE CLIENTES

1. CÓDIGO FUENTE

```
114      ' ====== ELIMINAR ======
115  @referencias
116  Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles btnEliminar.Click
117      If dgvResultados.SelectedRows.Count = 0 Then Exit Sub
118
119      Dim fila = dgvResultados.SelectedRows(0)
120
121      Dim frm As New EliminarCliente(
122          fila.Cells("cedula").Value.ToString(),
123          fila.Cells("nombre").Value.ToString(),
124          fila.Cells("telefono").Value.ToString(),
125          fila.Cells("correo").Value.ToString(),
126          fila.Cells("direccion").Value.ToString()
127      )
128
129      frm.ShowDialog()
130      BuscarCliente()
131  End Sub
132
133

Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles btnEliminar.Click
    ' Primera confirmación
    Dim result = MessageBox.Show("¿Estás seguro de eliminar al cliente con cédula " & cedulaCliente & "?",
        "Confirmar eliminación",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Warning)

    If result = DialogResult.Yes Then
        ' Segunda confirmación
        Dim finalConfirm = MessageBox.Show("¡Esta acción no se puede deshacer! ¿Seguro que deseas eliminar este cliente?",
            "Confirmación final",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning)

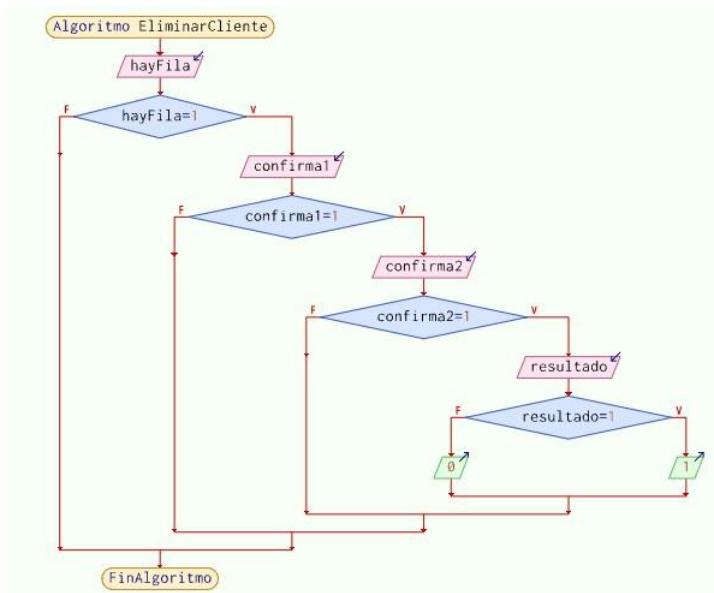
        If finalConfirm = DialogResult.Yes Then
            Try
                ' Eliminar de MongoDB
                Dim db = ConexionMongo.ObtenerConexion()
                Dim collection = db.GetCollection(Of BsonDocument)("Clientes")

                Dim filtro = New BsonDocument("cedula", cedulaCliente)
                collection.DeleteOne(filtro)

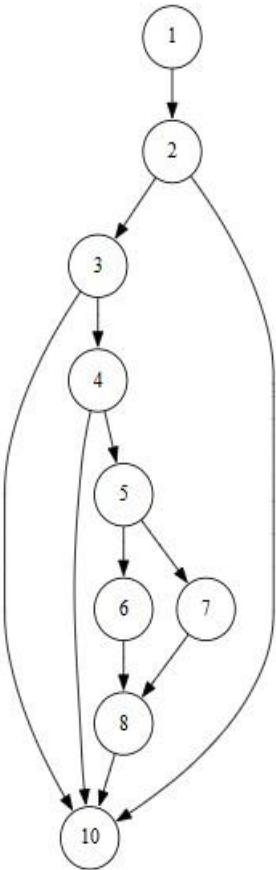
                MessageBox.Show("Cliente eliminado correctamente.", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)
                Me.Close()

            Catch ex As Exception
                MessageBox.Show("Error al eliminar: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End Try
        End If
    End If
End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 10	No hay fila seleccionada
R2	1 → 2 → 3 → 10	Fila seleccionada, primera confirmación = No
R3	1 → 2 → 3 → 4 → 10	Primera confirmación Sí, segunda confirmación = No
R4	1 → 2 → 3 → 4 → 5 → 6 → 8 → 10	Eliminación exitosa
R5	1 → 2 → 3 → 4 → 5 → 7 → 8 → 10	Error al eliminar cliente

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 10
- P (Número de nodos predicados): 4
- A (Número de aristas): 13

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 4 + 1 = 5$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 13 - 10 + 2 = 5$$

Resultado La complejidad ciclomática del requisito **Eliminar Cliente** es **5**, lo que indica que existen **cinco caminos básicos independientes** que deben ser probados para garantizar la cobertura completa del flujo de control.

DONDE

P: Número de nodos predicado

A: Número de aristas

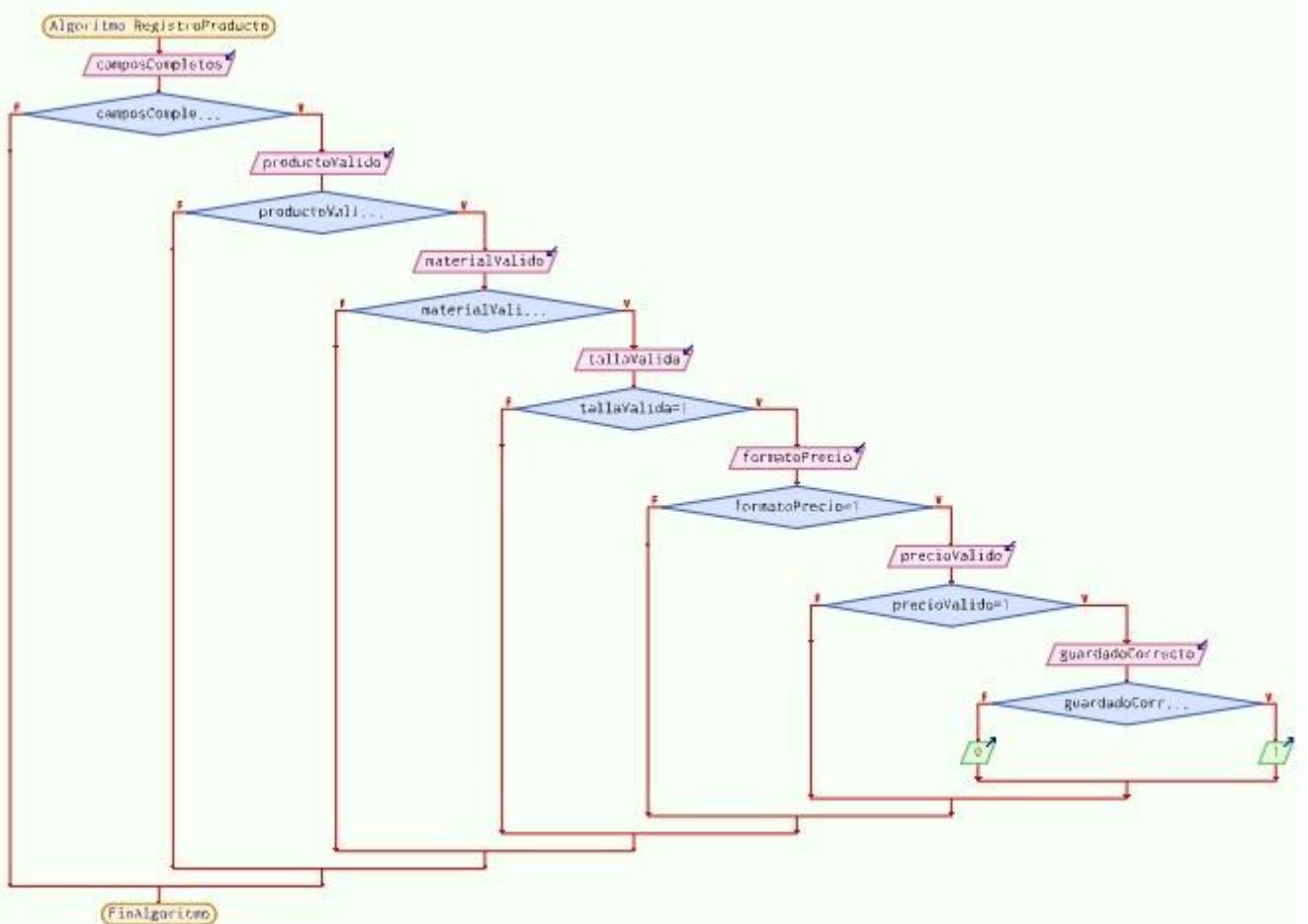
N: Número de nodos

RF N6^a REGISTROS DE PRODUCTOS

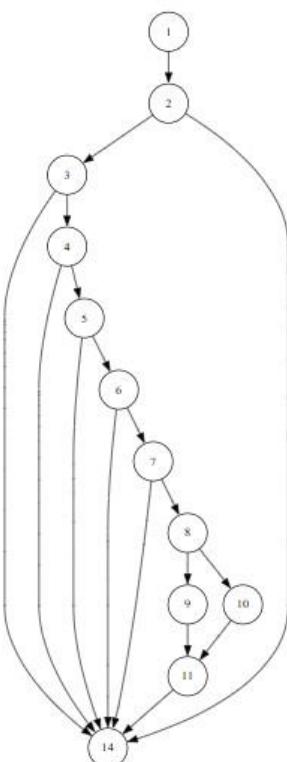
1. CÓDIGO FUENTE

```
55     ' Validación precio
56     If Not Regex.IsMatch(txtPrecioUnitario.Text.Trim, "^\d+(\.\d{1,2})?") Then
57         MessageBox.Show("El precio unitario debe ser un número válido, máximo 2 decimales")
58         txtPrecioUnitario.Focus()
59         Exit Sub
60     End If
61
62     Dim precio As Double
63     If Not Double.TryParse(txtPrecioUnitario.Text.Trim, precio) Then
64         MessageBox.Show("Ingrese un precio válido")
65         txtPrecioUnitario.Focus()
66         Exit Sub
67     End If
68
69     ' Guardar producto
70     Dim nuevoProducto As New Producto With {
71         .Producto = txtProducto.Text.Trim(),
72         .Talla = valorTalla,
73         .TipoMaterial = txtTipoMaterial.Text.Trim(),
74         .PrecioUnitario = precio
75     }
76
77     Try
78         productosCollection.InsertOne(nuevoProducto)
79         MessageBox.Show("Producto registrado correctamente")
80         LimpiarCampos()
81     Catch ex As Exception
82         MessageBox.Show("Error al guardar: " & ex.Message)
83     End Try
84 End Sub
85
86
87 ' ===== BOTÓN GUARDAR =====
88 0 referencias
89 Private Sub btnGuardar_Click(sender As Object, e As EventArgs) Handles btnGuardar.Click
90     ' Campos obligatorios
91     If txtProducto.Text.Trim = "" OrElse
92         txtTalla.Text.Trim = "" OrElse
93         txtTipoMaterial.Text.Trim = "" OrElse
94         txtPrecioUnitario.Text.Trim = "" Then
95             MessageBox.Show("Complete todos los campos")
96             Exit Sub
97         End If
98
99     ' Validación producto
100    If Not Regex.IsMatch(txtProducto.Text.Trim, "^[a-zA-Z\s]+") Then
101        MessageBox.Show("El nombre del producto solo debe contener letras y espacios")
102        txtProducto.Focus()
103        Exit Sub
104    End If
105
106    ' Validación tipo de material
107    If Not Regex.IsMatch(txtTipoMaterial.Text.Trim, "^[a-zA-Z\s]+") Then
108        MessageBox.Show("El tipo de material solo debe contener letras y espacios")
109        txtTipoMaterial.Focus()
110        Exit Sub
111    End If
112
113    ' Validación talla lógica
114    Dim valorTalla As String = txtTalla.Text.Trim().ToUpper()
115    Dim tallasValidas As String() = {"S", "M", "L", "XL", "XXL"}
116    Dim esNumeroValido As Boolean = Integer.TryParse(valorTalla, Nothing) AndAlso CInt(valorTalla) >= 30 AndAlso CInt(valorTalla) <
117        tallasValidas.Contains(valorTalla) AndAlso Not esNumeroValido Then
118            MessageBox.Show("Ingrese una talla válida (S, M, L, XL, XXL o número entre 30 y 50)")
119            txtTalla.Focus()
120            Exit Sub
121        End If
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 14	Campos incompletos
R2	1 → 2 → 3 → 14	Producto inválido
R3	1 → 2 → 3 → 4 → 14	Material inválido
R4	1 → 2 → 3 → 4 → 5 → 14	Talla inválida
R5	1 → 2 → 3 → 4 → 5 → 6 → 14	Precio formato inválido
R6	1 → 2 → 3 → 4 → 5 → 6 → 7 → 14	Precio no numérico
R7	1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 11 → 14	Registro exitoso
R8	1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 10 → 11 → 14	Error al guardar

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 14
- P (Número de nodos predicados): 7
- A (Número de aristas): 4

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 7 + 1 = 8$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 20 - 14 + 2 = 8$$

Resultado

La complejidad ciclomática del requisito **Registro de Productos** es **8**, por lo tanto existen **8 caminos básicos independientes** que deben probarse.

DONDE

P: Número de nodos predicado

A: Número de aristas

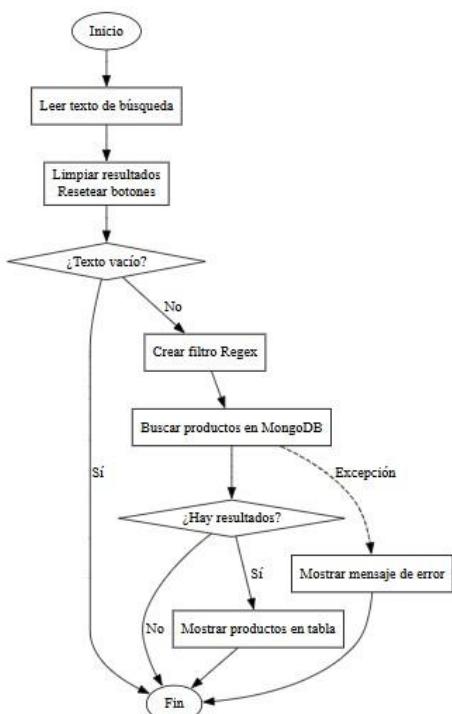
N: Número de nodos

RF N7ª BUSQUEDA DE PRODUCTOS

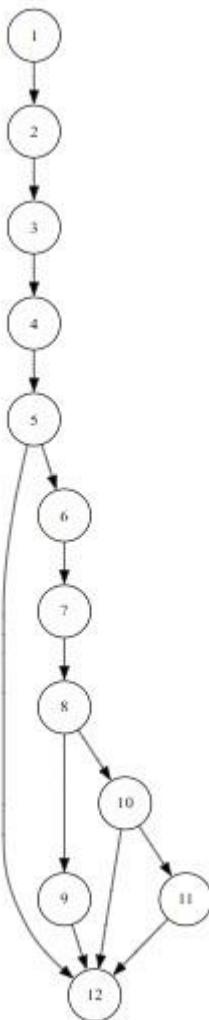
1. CÓDIGO FUENTE

```
42      | 3 referencias
43  Private Sub BuscarProducto()
44
45      Dim texto As String = txtBuscarProducto.Text.Trim()
46
47      dgvResultados.Rows.Clear()
48      productoSeleccionado = Nothing
49      btnEditar.Enabled = False
50      btnEliminar.Enabled = False
51
52      If texto = "" Then Exit Sub
53
54      Try
55          ' SIMILITUD (EMPIEZA CON / CONTIENE)
56          Dim filtro = Builders(Of Producto).Filter.Regex(
57              "producto",
58              New BsonRegularExpression(texto, "i")
59          )
60
61          Dim lista = productosCollection.Find(filtro).ToList()
62
63          For Each p In lista
64              dgvResultados.Rows.Add(
65                  p.Id,
66                  p.Producto,
67                  p.Talla,
68                  p.TipoMaterial,
69                  Math.Round(p.PrecioUnitario, 2)
70              )
71          Next
72
73      Catch ex As Exception
74          MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
75      End Try
76  End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de Nodos	Descripción
R1	1 – 2 – 3 – 4 – 5 – 12	El usuario no ingresa texto de búsqueda, el proceso termina sin consultar la base de datos.
R2	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12	El usuario ingresa texto válido, la consulta se ejecuta correctamente y se muestran los productos encontrados.
R3	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 12	El usuario ingresa texto válido, la consulta se ejecuta correctamente pero no se encuentran productos.
R4	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 12	El usuario ingresa texto válido pero ocurre un error durante la consulta a la base de datos y se muestra un mensaje de error.

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- **N (Número de nodos):** 12
- **P (Número de nodos predicados):** 4
- **A (Número de aristas):** 14

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 4 + 1 = 5$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 14 - 12 + 2 = 5$$

Resultado

La **complejidad ciclomática del requisito 7 (Búsqueda de productos por similitud)** es **5**, lo que indica que existen **cinco caminos básicos independientes** que deben ser probados mediante pruebas de caja blanca.

DONDE

P: Número de nodos predicado

A: Número de aristas

N: Número de nodos

RF N8ª MODIFICACION DE PRODUCTOS

1. CÓDIGO FUENTE

```
28     Private Sub btnGuardar_Click(sender As Object, e As EventArgs) Handles btnGuardar.Click
29
30         ' ===== VALIDAR CAMPOS VACÍOS =====
31         If txtProducto.Text.Trim() = "" OrElse
32             txtTalla.Text.Trim() = "" OrElse
33             txtTipoMaterial.Text.Trim() = "" OrElse
34             txtPrecioUnitario.Text.Trim() = "" Then
35                 MessageBox.Show("Complete todos los campos")
36                 Exit Sub
37             End If
38
39         ' ===== VALIDAR PRODUCTO =====
40         If Not Regex.IsMatch(txtProducto.Text.Trim(), "^[A-Za-z\s]+$") Then
41             MessageBox.Show("El producto solo debe contener letras y espacios")
42             txtProducto.Focus()
43             Exit Sub
44         End If
45
46         ' ===== VALIDAR TIPO MATERIAL =====
47         If Not Regex.IsMatch(txtTipoMaterial.Text.Trim(), "^[A-Za-z\s]+$") Then
48             MessageBox.Show("El tipo de material solo debe contener letras y espacios")
49             txtTipoMaterial.Focus()
50             Exit Sub
51         End If
52
53         ' ===== VALIDAR TALLA =====
54         Dim talla As String = txtTalla.Text.Trim().ToUpper()
55         Dim tallasValidas As String() = {"S", "M", "L", "XL", "XXL"}
56
57         Dim esTallaNumero As Boolean = Regex.IsMatch(talla, "\d+")
58         Dim esTallaLetra As Boolean = tallasValidas.Contains(talla)
59
60         If esTallaNumero Then
61             Dim tallaNum As Integer = CInt(talla)
62             If tallaNum < 30 OrElse tallaNum > 50 Then
63                 MessageBox.Show("La talla numérica debe estar entre 30 y 50")
64
65             txtTalla.Focus()
66             Exit Sub
67         End If
68         ElseIf Not esTallaLetra Then
69             MessageBox.Show("Ingrese una talla válida (S, M, L, XL, XXL o número entre 30 y 50)")
70             txtTalla.Focus()
71             Exit Sub
72         End If
73
74         ' ===== VALIDAR PRECIO =====
75         Dim precio As Decimal
76         If Not Decimal.TryParse(
77             txtPrecioUnitario.Text.Trim(),
78             NumberStyles.AllowDecimalPoint,
79             CultureInfo.InvariantCulture,
80             precio
81         ) Then
82             MessageBox.Show("Ingrese un precio válido")
83             txtPrecioUnitario.Focus()
84             Exit Sub
85         End If
86
87         precio = Math.Round(precio, 2)
88
89         ' ===== ACTUALIZAR EN MONGODB =====
90         Try
91             Dim filter = Builders(Of Producto).Filter.Eq(Function(p) p.Id, idProducto)
92
93             Dim update = Builders(Of Producto).Update _
94                 .Set(Function(p) p.Producto, txtProducto.Text.Trim()) _
95                 .Set(Function(p) p.Talla, talla) -
96                 .Set(Function(p) p.TipoMaterial, txtTipoMaterial.Text.Trim()) -
97                 .Set(Function(p) p.PrecioUnitario, Convert.ToDouble(precio))
98
99             productosCollection.UpdateOne(filter, update)
100
```

```

100     MessageBox.Show("Producto actualizado correctamente")
101     Me.Close()
102
103     Catch ex As Exception
104         MessageBox.Show("Error al actualizar: " & ex.Message)
105     End Try
106 End Sub
107
108     ' ----- CANCELAR -----
109     Private Sub btnCancelar_Click(sender As Object, e As EventArgs) Handles btnCancelar.Click
110         Me.Close()
111     End Sub
112
113     ' ----- BLOQUEO DE ENTRADAS -----
114
115     ' Producto: solo letras
116     Private Sub txtProducto_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtProducto.KeyPress
117         If Not Char.IsLetter(e.KeyChar) AndAlso
118             Not Char.IsWhiteSpace(e.KeyChar) AndAlso
119             Not Char.IsControl(e.KeyChar) Then
120                 e.Handled = True
121             End If
122     End Sub
123
124     ' Tipo material: solo letras
125     Private Sub txtTipoMaterial_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtTipoMaterial.KeyPress
126         If Not Char.IsLetter(e.KeyChar) AndAlso
127             Not Char.IsWhiteSpace(e.KeyChar) AndAlso
128             Not Char.IsControl(e.KeyChar) Then
129                 e.Handled = True
130             End If
131     End Sub

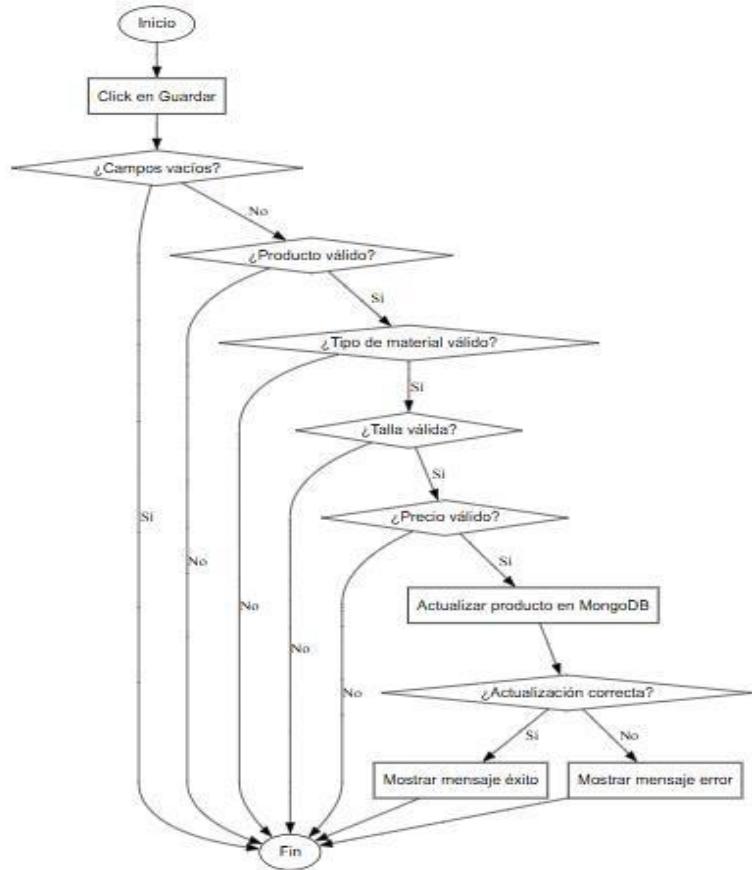
```

```

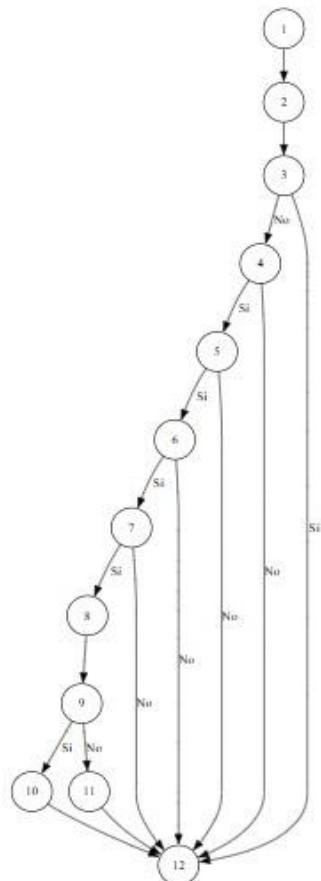
132
133     ' Talla: letras o números (no símbolos)
134     Private Sub txtTalla_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtTalla.KeyPress
135         If Not Char.IsLetterOrDigit(e.KeyChar) AndAlso
136             Not Char.IsControl(e.KeyChar) Then
137                 e.Handled = True
138             End If
139     End Sub
140
141     ' Precio: solo números y un punto
142     Private Sub txtPrecioUnitario_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtPrecioUnitario.KeyPress
143         Dim txt As TextBox = CType(sender, TextBox)
144
145         If Not Char.IsDigit(e.KeyChar) AndAlso
146             Not Char.IsControl(e.KeyChar) AndAlso
147             e.KeyChar <> "."c Then
148                 e.Handled = True
149             End If
150
151         ' Solo un punto decimal
152         If e.KeyChar = ".c" AndAlso txt.Text.Contains(".") Then
153             e.Handled = True
154         End If
155     End Sub
156
157 End Class

```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 12	Campos vacíos → el proceso termina sin actualizar
R2	1 → 2 → 3 → 4 → 12	Producto inválido → termina sin actualizar
R3	1 → 2 → 3 → 4 → 5 → 12	Tipo de material inválido → termina sin actualizar
R4	1 → 2 → 3 → 4 → 5 → 6 → 12	Talla inválida → termina sin actualizar
R5	1 → 2 → 3 → 4 → 5 → 6 → 7 → 12	Precio inválido → termina sin actualizar
R6	1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12	Todo correcto → actualización exitosa
R7	1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 11 → 12	Todo correcto → error en actualización

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 12
- P (Número de nodos predicados): 6
- A (Número de aristas): 14

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 6 + 1 = 7$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 14 - 12 + 2 = 4$$

Resultado

La complejidad ciclomática del REQ 8 es 7, lo que significa que existen **7 caminos básicos independientes** que debes probar para cobertura de caja blanca.

DONDE

P: Número de nodos predicado

A: Número de aristas

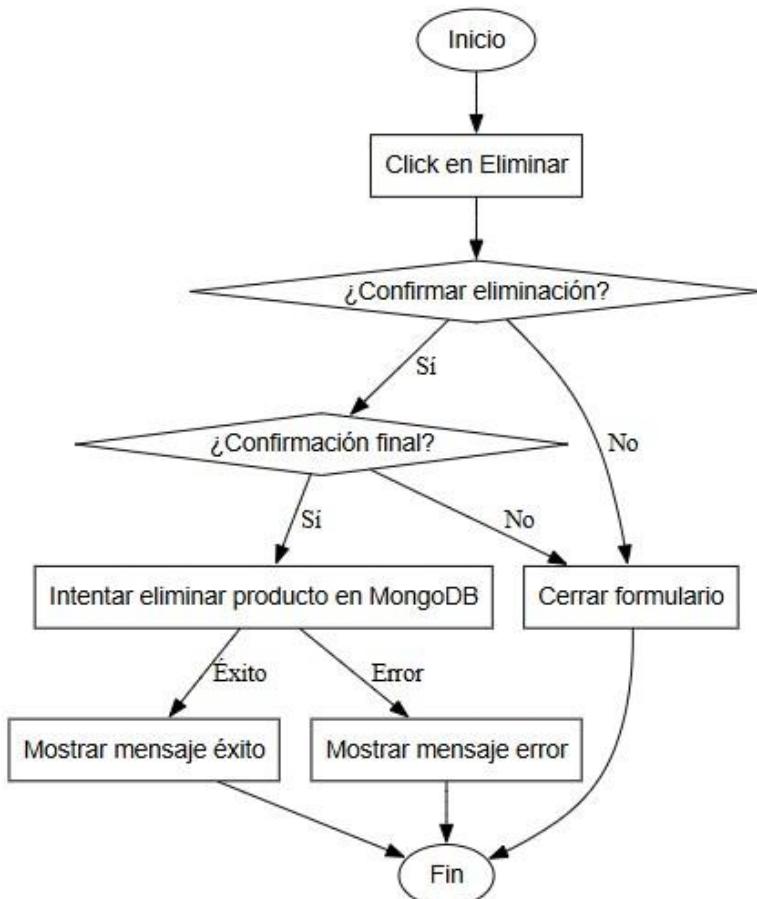
N: Número de nodos

RF N9ª ELIMINACION DE PRODUCTO

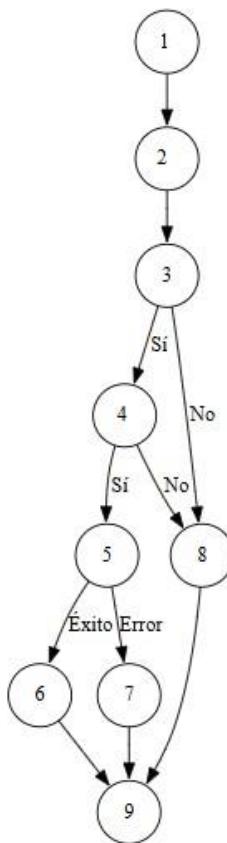
1. CÓDIGO FUENTE

```
28  Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles btnEliminar.Click
29      ' Primera confirmación
30      Dim result = MessageBox.Show("¿Estás seguro de eliminar el producto '" & txtNombre.Text & "'?", 
31          "Confirmar eliminación",
32          MessageBoxButtons.YesNo,
33          MessageBoxIcon.Warning)
34
35      If result = DialogResult.Yes Then
36          ' Segunda confirmación
37          Dim finalConfirm = MessageBox.Show("¡Esta acción no se puede deshacer! ¿Deseas eliminar este producto?", 
38              "Confirmación final",
39              MessageBoxButtons.YesNo,
40              MessageBoxIcon.Warning)
41
42          If finalConfirm = DialogResult.Yes Then
43              Try
44                  Dim db = ConexionMongo.ObtenerConexion()
45                  Dim collection = db.GetCollection(Of BsonDocument)("Productos")
46
47                  Dim filtro = Builders(Of BsonDocument).Filter.Eq(Of ObjectId)("_id", New ObjectId(idProducto))
48                  collection.DeleteOne(filtro)
49
50                  MessageBox.Show("Producto eliminado correctamente.", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)
51                  Me.Close()
52
53                  Catch ex As Exception
54                      MessageBox.Show("Error al eliminar: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
55                  End Try
56              End If
57          End If
58
59          ' ===== BOTÓN CANCELAR =====
60          0 referencias
61          Private Sub btnCancelar_Click(sender As Object, e As EventArgs) Handles btnCancelar.Click
62              Me.Close()
63          End Sub
64
65          0 referencias
66          Private Sub EliminarProducto_Load(sender As Object, e As EventArgs) Handles MyBase.Load
67              ' Inicialización extra si necesitas
68          End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 8 → 9	Usuario cancela en la primera confirmación → fin del proceso
R2	1 → 2 → 3 → 4 → 8 → 9	Usuario confirma primero, pero cancela en la segunda confirmación → fin del proceso
R3	1 → 2 → 3 → 4 → 5 → 6 → 9	Usuario confirma ambas y eliminación es exitosa → fin del proceso
R4	1 → 2 → 3 → 4 → 5 → 7 → 9	Usuario confirma ambas y ocurre error al eliminar → fin del proceso

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática permite determinar el número de caminos independientes del sistema.

Datos del grafo

- N (Número de nodos): 9
- P (Número de nodos predicados): 3
- A (Número de aristas): 10

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 3 + 1 = 4$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 10 - 9 + 2 = 3$$

Resultado

La complejidad ciclomática del REQ 9 es 7, lo que significa que existen **7 caminos básicos independientes** que debes probar para cobertura de caja blanca.

DONDE

P: Número de nodos predicado

A: Número de aristas

N: Número de nodos

Prueba caja blanca

RF N°: REQ010 Envío de Catalogo por correo

```
0 referencias
Private Sub btnEnviarCatalogo_Click(sender As Object, e As EventArgs) _
Handles btnEnviarCatalogo.Click

    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12

    Try
        If dgvResultados.SelectedRows.Count = 0 Then
            Throw New Exception("Seleccione un cliente.")
        End If

        Dim correoCliente As String =
            dgvResultados.SelectedRows(0).Cells("correo").Value.ToString().Trim()

        If correoCliente = "" Then
            Throw New Exception("El cliente no tiene correo.")
        End If

        ' ===== PDF SIEMPRE NUEVO =====
        Dim rutaPDF As String = Path.Combine(
            Path.GetTempPath(),
            "Catalogo_" & DateTime.Now.Ticks & ".pdf"
        )

        ModuloPDF.GenerarCatalogoPDF(rutaPDF)

        If Not File.Exists(rutaPDF) Then
            Throw New Exception("No se pudo generar el catálogo.")
        End If

        ' ===== CORREO =====
        Using mensaje As New MailMessage()
            mensaje.From = New MailAddress("kevin1mic2020@gmail.com")
            mensaje.To.Add(correoCliente)
            mensaje.Subject = "Catálogo de Productos - MarcallaTex"
            mensaje.Body = "Adjuntamos nuestro catálogo de productos actualizado."
            mensaje.Attachments.Add(New Attachment(rutaPDF))

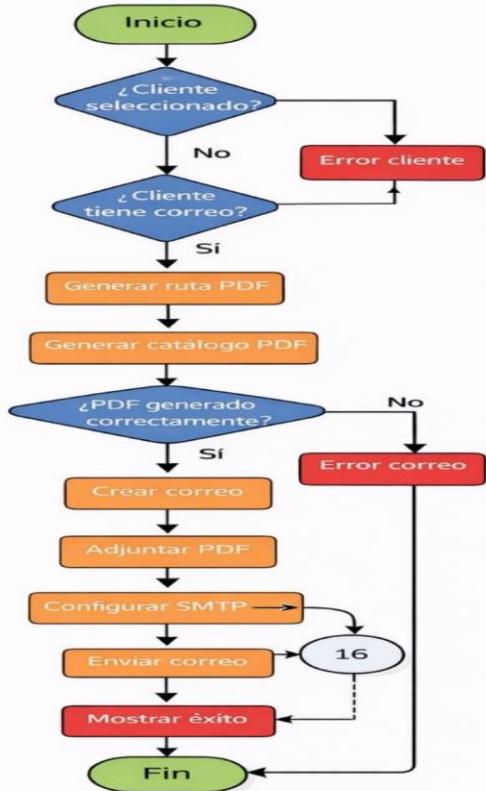
            Using smtp As New SmtpClient("smtp.gmail.com", 587)
                smtp.EnableSsl = True
                smtp.UseDefaultCredentials = False
                smtp.Credentials = New NetworkCredential(
                    "kevin1mic2020@gmail.com",
                    "uvjlonhyapgrxidk"
                )

                smtp.Send(mensaje)
            End Using
        End Using

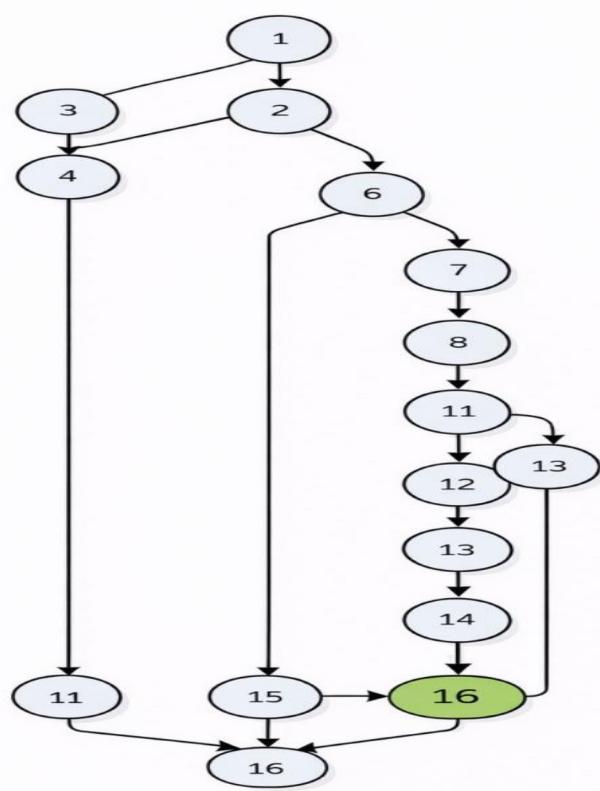
        MessageBox.Show("Catálogo enviado correctamente ✅",
            "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)

    Catch ex As Exception
        MessageBox.Show(ex.Message,
            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. Diagrama de grafo



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 16	No hay cliente seleccionado
R2	1 → 2 → 4 → 5 → 16	Cliente sin correo
R3	1 → 2 → 4 → 6 → 7 → 8 → 9 → 16	Error al generar el PDF
R4	1 → 2 → 4 → 6 → 7 → 8 → 10 → 11 → 12 → 13 → 14 → 16	Envío exitoso del catálogo
R5	1 → 2 → 4 → 6 → 7 → 8 → 10 → 11 → 12 → 13 → 15 → 16	Error durante el envío (excepción)

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática del Grafo de Flujo se calcula para determinar el número de **caminos independientes** del sistema.

Datos del grafo

- **N (Número de nodos):** 16
- **P (Número de nodos predicado):** 4
 - Nodo 2 → Cliente seleccionado
 - Nodo 4 → Correo válido
 - Nodo 8 → PDF generado
 - Nodo 13 → Resultado del envío
- **A (Número de aristas):** 19

Cálculo

Fórmula 1

$$V(G) = P + 1 \\ V(G) = 4 + 1 = 5$$

Fórmula 2

$$V(G) = A - N + 2 \\ V(G) = 19 - 16 + 2 = 5$$

Resultado

La **complejidad ciclomática del proceso “Enviar Catálogo por Correo” es 5**, lo que indica que existen **cinco caminos básicos independientes** que deben ser considerados en las pruebas del sistema.

RF N^a: REQ011- Generación de Proformas

1. CÓDIGO FUENTE

```
1 referencia
Private Sub ExportarAPDF(contentPanel As FlowLayoutPanel, productos As List(Of DetalleProducto))
    Try
        Dim saveDialog As New SaveFileDialog()
        saveDialog.Filter = "Archivos PDF (*.pdf)|*.pdf"
        saveDialog.FileName = $"Proforma_{clienteSeleccionado.Nombre.Replace(" ", "_")}{DateTime.Now:yyyyMMdd_HHnmss}.pdf"
        If saveDialog.ShowDialog() = DialogResult.OK Then
            rutaPDFGenerado = saveDialog.FileName
            Dim documento As New Document(PageSize.A4, 36, 36, 36, 36)
            Dim writer As PdfWriter = PdfWriter.GetInstance(documento, New FileStream(saveDialog.FileName, FileMode.Create))
            documento.Open()

            Dim fuenteSubtitulo As New PdfFont(PdfFont.FontFamily.HELVETICA, 12, PdfFont.BOLD)
            Dim fuenteNormal As New PdfFont(PdfFont.FontFamily.HELVETICA, 10)
            Dim fLabel As New PdfFont(PdfFont.FontFamily.HELVETICA, 10, PdfFont.BOLD)
            Dim fData As New PdfFont(PdfFont.FontFamily.HELVETICA, 10, PdfFont.NORMAL)

            Dim rutaLogo As String = Path.Combine(Application.StartupPath, "logo.png")
            If File.Exists(rutaLogo) Then
                Try
                    Dim imagen As iTextSharp.text.Image = iTextSharp.text.Image.GetInstance(rutaLogo)
                    imagen.ScaleToFit(180.0F, 80.0F)
                    imagen.Alignment = iTextSharp.text.Image.ALIGN_CENTER
                    documento.Add(imagen)
                Catch ex As Exception
                End Try
            End If

            Dim espacio As New Paragraph(" ")
            espacio.SpacingAfter = 5
            documento.Add(espacio)

            Dim linea As New PdfPTable(1)
            linea.WidthPercentage = 100
            Dim celdaLinea As New PdfPCell(New Phrase(""))
            celdaLinea.Border = iTextSharp.text.Rectangle.BOTTOM_BORDER
            celdaLinea.BorderWidthBottom = 2.0F
            celdaLinea.FixedHeight = 5
            celdaLinea.BorderColorBottom = BaseColor.BLACK
            celdaLinea.Padding = 0
            linea.AddCell(celdaLinea)
```

```
celdaLinea.Padding = 0
linea.AddCell(celdaLinea)
documento.Add(linea)

FontFactory.RegisterDirectory("C:\Windows\Fonts")
Dim fuenteTitulo As iTextSharp.text.Font = FontFactory.GetFont("Arial", 20, iTextSharp.text.Font.BOLD, BaseColor.BLACK)
If fuenteTitulo.BaseFont Is Nothing Then
    fuenteTitulo = New PdfFont(PdfFont.FontFamily.HELVETICA, 20, PdfFont.BOLD, BaseColor.BLACK)
End If

Dim titulo As New Paragraph("PROFORMA DE COTIZACIÓN", fuenteTitulo)
titulo.Alignment = Element.ALIGN_CENTER
titulo.SpacingBefore = 10
titulo.SpacingAfter = 20
documento.Add(titulo)

Dim infoCliente As New PdfPTable(2)
infoCliente.WidthPercentage = 100
infoCliente.SetWidths({1.2F, 0.8F})

Dim celdaIzq As New PdfPCell()
celdaIzq.Border = iTextSharp.text.Rectangle.NO_BORDER
celdaIzq.Padding = 8
celdaIzq.BackgroundColor = New BaseColor(245, 245, 245)

Dim pCliente As New Paragraph()
pCliente.Add(New Chunk("CLIENTE: ", fLabel))
pCliente.Add(New Chunk(clienteSeleccionado.Nombre, fData))
celdaIzq.AddElement(pCliente)
Dim pCedula As New Paragraph()
pCedula.Add(New Chunk("CÉDULA/RUC: ", fLabel))
pCedula.Add(New Chunk(clienteSeleccionado.Cedula, fData))
celdaIzq.AddElement(pCedula)
Dim pTelf As New Paragraph()
pTelf.Add(New Chunk("TELÉFONO: ", fLabel))
pTelf.Add(New Chunk(clienteSeleccionado.Telefono, fData))
celdaIzq.AddElement(pTelf)
Dim pCorreo As New Paragraph()
pCorreo.Add(New Chunk("CORREO: ", fLabel))
pCorreo.Add(New Chunk(clienteSeleccionado.Correo, fData))
celdaIzq.AddElement(pCorreo)
Dim pDir As New Paragraph()
pDir.Add(New Chunk("DIRECCIÓN: ", fLabel))
pDir.Add(New Chunk(clienteSeleccionado.Direccion, fData))
```

```

        <-- PROTOTIPO
pDir.Add(New Chunk("DIRECCION: ", fLabel))
pDir.Add(New Chunk(clienteSeleccionado.Direccion, fData))
celdaIzq.AddElement(pDir)

Dim celdaDer As New PdfPCell()
celdaDer.Border = iTextSharp.text.Rectangle.BOX
celdaDer.Padding = 8
celdaDer.BackgroundColor = New BaseColor(245, 245, 245)
celdaDer.VerticalAlignment = Element.ALIGN_MIDDLE

Dim pFecha As New Paragraph()
pFecha.Alignment = Element.ALIGN_LEFT
pFecha.Add(New Chunk("FECHA: ", fLabel))
pFecha.Add(New Chunk(DateTime.Now.ToString("dd/MM/yyyy"), fData))
celdaDer.AddElement(pFecha)
Dim pHora As New Paragraph()
pHora.Alignment = Element.ALIGN_LEFT
pHora.Add(New Chunk("HORA: ", fLabel))
pHora.Add(New Chunk(DateTime.Now.ToString("HH:mm"), fData))
celdaDer.AddElement(pHora)

infoCliente.AddCell(celdaIzq)
infoCliente.AddCell(celdaDer)
infoCliente.SpacingAfter = 20
documento.Add(infoCliente)

Dim tablaProductos As New PdfPTable(6)
tablaProductos.WidthPercentage = 100
tablaProductos.SetWidths({0.6F, 2.0F, 2.5F, 0.8F, 1.0F, 1.0F})

Dim encabezados() As String = {"#", "PRODUCTO", "DESCRIPCIÓN", "CANT.", "PRECIO UNIT.", "SUBTOTAL"}
For Each enc In encabezados
    Dim celda As New PdfPCell(New Phrase(enc, New PdfFont(PdfFont.FontFamily.HELVETICA, 9, PdfFont.BOLD, BaseColor.WHITE)))
    celda.BackgroundColor = New BaseColor(52, 73, 94)
    celda.Horizontal.setAlignment = Element.ALIGN_CENTER
    celda.Padding = 8
    celda.Border = iTextSharp.text.Rectangle.BOX
    tablaProductos.addCell(celda)
Next

Dim itemNum As Integer = 1
For Each prod In productos
    Dim celdaNum As New PdfPCell(New Phrase(itemNum.ToString(), fuenteNormal))
    celdaNum.Horizontal.setAlignment = Element.ALIGN_CENTER

```

```

    Dim celdaNum As New PdfPCell(New Phrase(itemNum.ToString(), fuenteNormal))
    celdaNum.Horizontal.setAlignment = Element.ALIGN_CENTER
    celdaNum.Padding = 5
    tablaProductos.addCell(celdaNum)
    Dim celdaProd As New PdfPCell(New Phrase(prod.Producto, fuenteNormal))
    celdaProd.Horizontal.setAlignment = Element.ALIGN_LEFT
    celdaProd.Padding = 5
    tablaProductos.addCell(celdaProd)
    Dim textoDesc As String = If(prod.Descripcion Is Nothing, "", prod.Descripcion)
    Dim celdaDesc As New PdfPCell(New Phrase(textoDesc, fuenteNormal))
    celdaDesc.Horizontal.setAlignment = Element.ALIGN_LEFT
    celdaDesc.Padding = 5
    tablaProductos.addCell(celdaDesc)
    Dim celdaCant As New PdfPCell(New Phrase(prod.Cantidad.ToString(), fuenteNormal))
    celdaCant.Horizontal.setAlignment = Element.ALIGN_CENTER
    celdaCant.Padding = 5
    tablaProductos.addCell(celdaCant)
    Dim celdaPrecio As New PdfPCell(New Phrase("$ " & prod.Precio.ToString("0.00"), fuenteNormal))
    celdaPrecio.Horizontal.setAlignment = Element.ALIGN_RIGHT
    celdaPrecio.Padding = 5
    tablaProductos.addCell(celdaPrecio)
    Dim celdaSubtotal As New PdfPCell(New Phrase("$ " & prod.Subtotal.ToString("0.00"), fuenteNormal))
    celdaSubtotal.Horizontal.setAlignment = Element.ALIGN_RIGHT
    celdaSubtotal.Padding = 5
    tablaProductos.addCell(celdaSubtotal)
    itemNum += 1
Next

tablaProductos.SpacingAfter = 20
documento.Add(tablaProductos)

Dim lineaInferior As New PdfPTable(1)
lineaInferior.WidthPercentage = 100
Dim celdaLineaInf As New PdfPCell(New Phrase(""))
celdaLineaInf.Border = iTextSharp.text.Rectangle.BOTTOM_BORDER
celdaLineaInf.BorderWidthBottom = 2.0F
celdaLineaInf.FixedHeight = 18
celdaLineaInf.BorderColorBottom = BaseColor.BLACK
celdaLineaInf.Padding = 0
lineaInferior.addCell(celdaLineaInf)
documento.Add(lineaInferior)

Dim tablaTotales As New PdfPTable(2)
tablaTotales.WidthPercentage = 100

```

```

Dim lineaInferior As New PdfPTable(1)
lineaInferior.WidthPercentage = 100
Dim celdalineaInf As New PdfPCell(New Phrase(""))
celdalineaInf.Border = iTextSharp.text.Rectangle.BOTTOM_BORDER
celdalineaInf.BorderWidthBottom = 2.0F
celdalineaInf.FixedHeight = 10
celdalineaInf.BorderColorBottom = BaseColor.BLACK
celdalineaInf.Padding = 8
lineaInferior.AddCell(celdalineaInf)
documento.Add(lineaInferior)

Dim tablaTotales As New PdfPTable(2)
tablaTotales.WidthPercentage = 100
tablaTotales.SetWidths({2.0F, 1.0F})
Dim celdaObs As New PdfPCell()
celdaObs.Border = iTextSharp.text.Rectangle.NO_BORDER
celdaObs.AddElement(New Paragraph("OBSERVACIONES:", fuenteSubtitulo))
Dim observacionesTexto As String = If(String.IsNullOrEmpty(txtObservaciones.Text), "Ninguna", txtObservaciones.Text)
celdaObs.AddElement(New Paragraph(observacionesTexto, fuenteNormal))
tablaTotales.AddCell(celdaObs)

Dim subtotal As Decimal = productos.Sum(Function(p) p.Subtotal)
Dim abono As Decimal = 0
Decimal.TryParse(txtAbono.Text.Replace(",", ".") , NumberStyles.Any, CultureInfo.InvariantCulture, abono)
Dim total As Decimal = subtotal - abono

Dim celdaTotales As New PdfPCell()
celdaTotales.Border = iTextSharp.text.Rectangle.NO_BORDER
celdaTotales.HorizontalAlignment = Element.ALIGN_RIGHT
celdaTotales.AddElement(New Paragraph($"Subtotal: ${subtotal:0.00}", fuenteSubtitulo))
celdaTotales.AddElement(New Paragraph($"Abono: ${abono:0.00}", fuenteSubtitulo))
celdaTotales.AddElement(New Paragraph(" "))
Dim parrafoTotal As New Paragraph($"Total a Pagar: ${total:0.00}" , New PdfFont(PdfFont.FontFamily.HELVETICA, 14, PdfFont.BOLD, BaseColor.RED))
parrafoTotal.Alignment = Element.ALIGN_RIGHT
celdaTotales.AddElement(parrafoTotal)
tablaTotales.AddCell(celdaTotales)
tablaTotales.SpacingAfter = 30
documento.Add(tablaTotales)

documento.Add(New Paragraph(" "))
documento.Add(New Paragraph(" "))
documento.Add(New Paragraph(" "))
Dim firma As New Paragraph(" ", fuenteNormal)

```

	<pre> documento.Add(New Paragraph(" ")) documento.Add(New Paragraph(" ")) documento.Add(New Paragraph(" ")) Dim firma As New Paragraph(" ", fuenteNormal) firma.Alignment = Element.ALIGN_CENTER documento.Add(firma) Dim textoFirma As New Paragraph("FIRMA Y SELLO", fuenteNormal) textoFirma.Alignment = Element.ALIGN_CENTER documento.Add(textoFirma) documento.Close() writer.Close() rutaPDFGenerado = saveDialog.FileName MessageBox.Show(\$"PDF generado exitosamente en:{vbCrLf}{saveDialog.FileName}" , "Exito", MessageBoxButtons.OK, MessageBoxIcon.Information) End If Catch ex As Exception MessageBox.Show(\$"Error al generar PDF: {ex.Message}" , "Error", MessageBoxButtons.OK, MessageBoxIcon.Error) End Try End Sub </pre>
--	--

```

0 referencias
Private Sub btnVistaPrevia_Click(sender As Object, e As EventArgs) Handles btnVistaPrevia.Click
    ' 1. Validaciones
    If clienteSeleccionado Is Nothing Then
        MessageBox.Show("Seleccione un cliente válido.", "Cliente", MessageBoxButtons.OK, MessageBoxIcon.Warning)
        Exit Sub
    End If

    ' 2. Recolección de datos
    Dim productosList As New List(Of DetalleProducto)
    For Each fila As DataGridViewRow In dgvDetalle.Rows
        If Not fila.IsNewRow AndAlso fila.Cells(0).Value IsNot Nothing AndAlso Not String.IsNullOrWhiteSpace(fila.Cells(0).Value.ToString()) Then
            Dim cant As Integer = 0
            Dim precio As Decimal = 0
            Dim subT As Decimal = 0
            Dim desc As String = fila.Cells(1).Value?.ToString()
            Integer.TryParse(fila.Cells(2).Value?.ToString(), cant)
            Decimal.TryParse(fila.Cells(3).Value?.ToString()?.Replace(",","."), NumberStyles.Any, CultureInfo.InvariantCulture, precio)
            Decimal.TryParse(fila.Cells(4).Value?.ToString()?.Replace(",","."), NumberStyles.Any, CultureInfo.InvariantCulture, subT)

            productosList.Add(New DetalleProducto With {
                .Producto = fila.Cells(0).Value.ToString(),
                .Descripcion = desc,
                .Cantidad = cant,
                .Precio = precio,
                .Subtotal = subT
            })
        End If
    Next

    If productosList.Count = 0 Then
        MessageBox.Show("Agregue al menos un producto.", "Productos", MessageBoxButtons.OK, MessageBoxIcon.Warning)
        Exit Sub
    End If

    ' 3. Configuración de la Ventana
    Dim visorForm As New Form() With {
        .Text = "Vista Previa de Proforma",
        .Size = New Size(950, 800),
        .StartPosition = FormStartPosition.CenterScreen,
        .BackColor = Color.White ' Fondo del formulario BLANCO
    }

```

```

    }

    ' Panel de fondo (Donde va la hoja) - AHORA BLANCO
    Dim panelPreview As New Panel() With {
        .Dock = DockStyle.Fill,
        .AutoScroll = True,
        .BackColor = Color.White ' <--- CAMBIO AQUÍ: FONDO BLANCO
    }

    ' La "Hoja" de papel
    Dim contentPanel As New FlowLayoutPanel() With {
        .FlowDirection = FlowDirection.TopDown,
        .WrapContents = False,
        .AutoSize = True,
        .AutoSizeMode = AutoSizeMode.GrowAndShrink,
        .Padding = New Padding(40),
        .BackColor = Color.White,
        .BorderStyle = BorderStyle.FixedSingle ' Borde fino para distinguir la hoja del fondo blanco
    }

    ' Generar contenido
    GenerarContenidoProforma(contentPanel, productosList)
    panelPreview.Controls.Add(contentPanel)

    ' Lógica de Centrado
    Dim CentrarHoja = Sub()
        Dim x As Integer = (panelPreview.ClientSize.Width - contentPanel.Width) \ 2
        If x < 10 Then x = 10
        contentPanel.Location = New Point(x, 10)
    End Sub
    AddHandler panelPreview.Resize, Sub(s, ev) CentrarHoja()
    AddHandler visorForm.Shown, Sub(s, ev) CentrarHoja()

    ' 4. Panel de Botones (Inferior)
    Dim panelBotones As New Panel() With {
        .Dock = DockStyle.Bottom,
        .Height = 70,
        .BackColor = Color.FromArgb(52, 73, 94) ' Azul oscuro para el pie
    }

    ' ---- BOTÓN VERDE (GENERAR) ----
    Dim btnAcciónPDF As New Button() With {
        .Text = "GENERAR / IMPRIMIR PDF",
        .Size = New Size(200, 50)
    }

```

```

        .BackColor = Color.FromArgb(46, 204, 113),
        .ForeColor = Color.White,
        .Font = New WinFont("Segoe UI", 11, FontStyle.Bold),
        .FlatStyle = FlatStyle.Flat,
        .Cursor = Cursors.Hand
    }
}
btnAccionPDF.FlatAppearance.BorderSize = 0
AddHandler btnAccionPDF.Click, Sub(s, ev) ExportarAPDF(contentPanel, productosList)

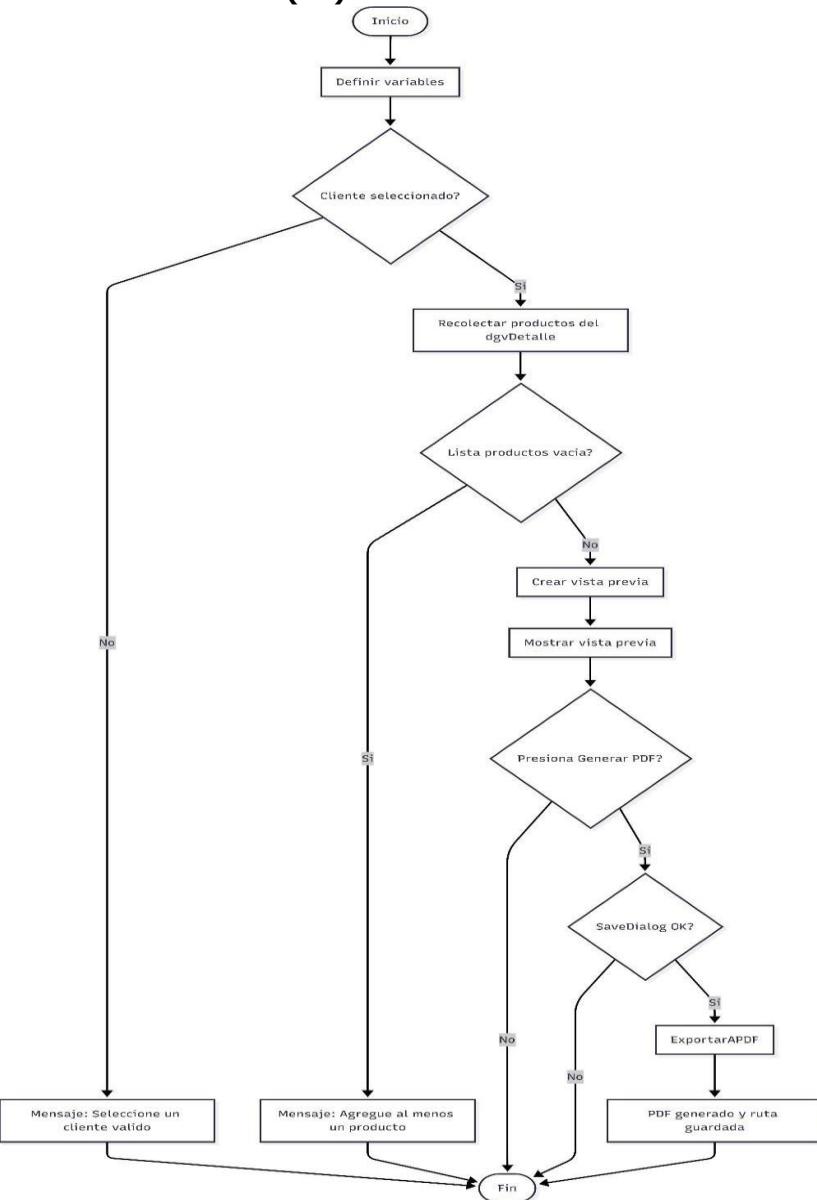
' ---- BOTÓN ROJO (CERRAR) ---
Dim btnCerrar As New Button() With {
    .Text = "CERRAR",
    .Size = New Size(150, 45),
    .Location = New Point(visorForm.Width - 190, 12),
    .Anchor = AnchorStyles.Top Or AnchorStyles.Right, ' Para que se mueva al maximizar
    .BackColor = Color.FromArgb(231, 76, 60),
    .ForeColor = Color.White,
    .Font = New WinFont("Segoe UI", 11, FontStyle.Bold),
    .FlatStyle = FlatStyle.Flat,
    .Cursor = Cursors.Hand
}
btnCerrar.FlatAppearance.BorderSize = 0
AddHandler btnCerrar.Click, Sub(s, ev) visorForm.Close()

' Agregar botones al panel
panelBotones.Controls.AddRange({btnAccionPDF, btnCerrar})

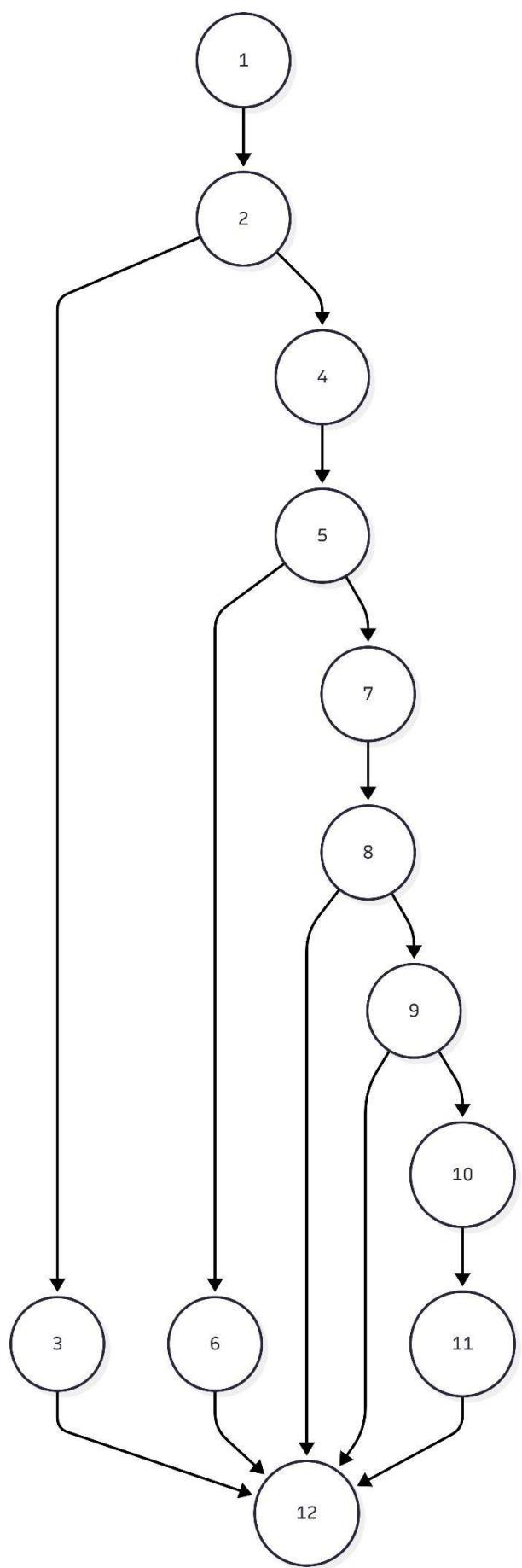
' Agregar paneles al formulario
visorForm.Controls.Add(panelPreview)
visorForm.Controls.Add(panelBotones)
visorForm.ShowDialog()
End Sub

```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



- 1 = Inicio
- 2 = Validar cliente seleccionado
- 3 = Mensaje cliente invalido
- 4 = Recolectar productos
- 5 = Validar lista productos vacía
- 6 = Mensaje sin productos
- 7 = Crear vista previa
- 8 = Usuario decide generar PDF
- 9 = SaveDialog OK?
- 10 = Exportar PDF
- 11 = Guardar rutaPDFGenerado / mensaje éxito
- 12 = Fin

4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 12	Flujo correcto: genera PDF
R2	1 → 2 → 3 → 12	No hay cliente seleccionado
R3	1 → 2 → 4 → 5 → 6 → 12	No hay productos
R4	1 → 2 → 4 → 5 → 7 → 8 → 12	No presiona generar PDF
R5	1 → 2 → 4 → 5 → 7 → 8 → 9 → 12	Cancela el SaveDialog

5. COMPLEJIDAD CICLOMÁTICA

Datos del grafo (según nuestro GF)

- N (Número de nodos): 12
- P (Número de nodos predicados): 4
(son los que tienen 2 salidas: 2, 5, 8, 9)
- A (Número de aristas): 15

Cálculo

Formula 1:

$$V(G) = P + 1 = 4 + 1 = 5$$

Formula 2:

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

Resultado: La complejidad ciclomática es 5 (existen 5 caminos básicos).

Prueba caja blanca

RF N°: REQ012 Envío de Proformas por Correo

1. CÓDIGO FUENTE

```
Public Sub EnviarCorreoPDF(rutaPDF As String)
    Try
        ' Validar PDF
        If String.IsNullOrEmpty(rutaPDFGenerado) OrElse Not File.Exists(rutaPDFGenerado) Then
            MessageBox.Show("Primero genere el PDF de la proforma.",
                           "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Exit Sub
        End If

        ' Validar cliente y correo
        If clienteSeleccionado Is Nothing OrElse String.IsNullOrEmpty(clienteSeleccionado.Correo) Then
            MessageBox.Show("El cliente seleccionado no tiene correo válido.",
                           "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Exit Sub
        End If

        ' Confirmar envío
        Dim correoCliente As String = clienteSeleccionado.Correo
        If MessageBox.Show(
            "¿Desea enviar el PDF al correo " & correoCliente & "?",
            "Confirmar envío",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Question
        ) = DialogResult.No Then
            Exit Sub
        End If

        ' Crear mensaje
        Dim correo As New MailMessage()
        correo.From = New MailAddress("tuemail@gmail.com", "MarcallaTex")
        correo.To.Add(correoCliente)
        correo.Subject = "Proforma de Cotización"
        correo.Body =
            "Estimado/a " & clienteSeleccionado.Nombre & "," & vbCrLf & vbCrLf &
            "Adjunto encontrará su proforma de cotización." & vbCrLf &
            "Saludos."
        correo.Attachments.Add(New Attachment(rutaPDFGenerado))

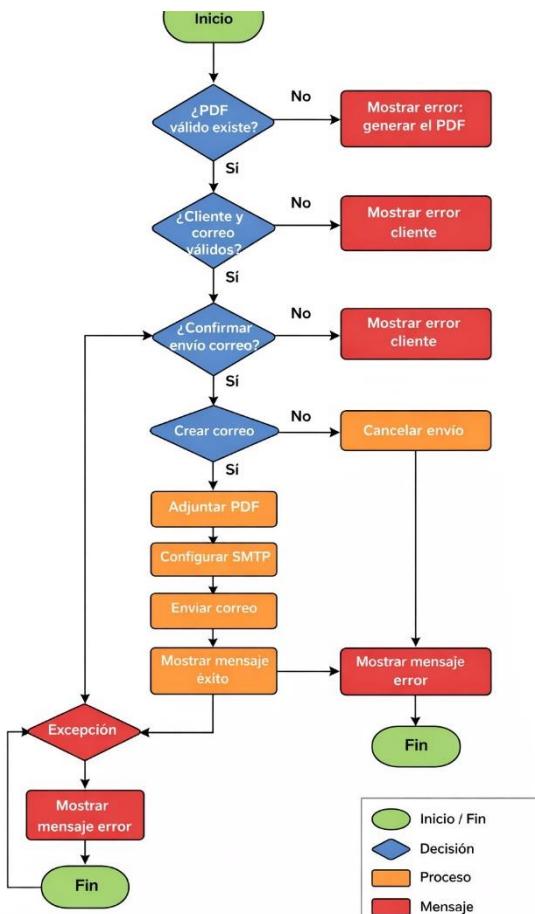
        ' Configuración SMTP
        Dim smtp As New SmtpClient("smtp.gmail.com", 587)
        smtp.EnableSsl = True
        smtp.Credentials = New System.Net.NetworkCredential(
            "kevin1mic2020@gmail.com",
            "uvjlonhyapgrxidk"
        )

        ' Enviar
        smtp.Send(correo)

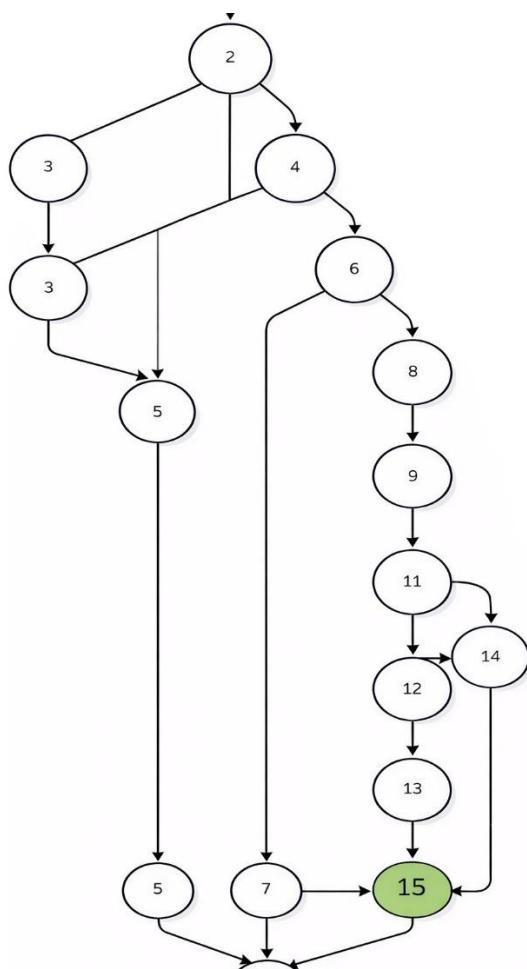
        MessageBox.Show("Correo enviado correctamente ✅",
                       "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)

    Catch ex As Exception
        MessageBox.Show("Error al enviar correo: " & ex.Message,
                       "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1 → 2 → 3 → 15	PDF no generado o inválido, el proceso se detiene
R2	1 → 2 → 4 → 5 → 15	Cliente no seleccionado o correo inválido
R3	1 → 2 → 4 → 6 → 7 → 15	Usuario cancela el envío del correo
R4	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 12 → 13 → 15	Envío exitoso del PDF por correo
R5	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 14 → 15	Error durante el envío (excepción SMTP u otro error)

5. COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática del Grafo de Flujo se calcula para determinar el número de caminos independientes del sistema.

Datos del grafo

- **N (Número de nodos):** 15
- **P (Número de nodos predicados):** 4
 - Nodo 2 → Validar PDF
 - Nodo 4 → Validar cliente
 - Nodo 6 → Confirmar envío
 - Nodo 11 → Resultado del envío (éxito / excepción)
- **A (Número de aristas):** 18

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 4 + 1 = 5$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 18 - 15 + 2 = 5$$

Resultado

La complejidad ciclomática del proceso “Enviar PDF por correo” es 5, lo que indica que existen **cinco caminos básicos independientes** que deben ser considerados para pruebas y validación del sistema.

Prueba caja blanca

RF N°: REQ013 HISTORIAL DE PROFORMAS

1. CÓDIGO FUENTE

```
0 referencias
Private Sub btnGuardarDrive_Click(sender As Object, e As EventArgs) Handles btnGuardarDrive.Click
    If String.IsNullOrEmpty(rutaPDFGenerado) Then
        MessageBox.Show("No se ha generado ninguna proforma todavía." & vbCrLf & "Por favor, presione 'GENERAR / IMPRIMIR PDF' en la Vista Previa.", "Falta PDF", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Exit Sub
    End If
    If Not File.Exists(rutaPDFGenerado) Then
        MessageBox.Show("El archivo PDF indicado no existe en el disco.", "Error de Archivo", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Exit Sub
    End If
    If clienteSeleccionado Is Nothing Then
        MessageBox.Show("No hay un cliente seleccionado.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Exit Sub
    End If

    Me.Cursor = Cursors.WaitCursor
    btnGuardarDrive.Enabled = False
    Try
        Dim carpetaRaizId As String = "1EbKQ9taMWrOH8gjVoxsPa8VILsUN580y"
        Dim nombreClienteFolder As String = clienteSeleccionado.Nombre.Trim().Replace("/", "-").Replace("\\", "-")
        Dim carpetaClienteId As String = GoogleDriveHelper.ObtenerOCrearCarpet(a(nombreClienteFolder, carpetaRaizId))
        GoogleDriveHelper.SubirArchivo(rutaPDFGenerado, carpetaClienteId)
        MessageBox.Show($"Proforma subida exitosamente.", "Google Drive", MessageBoxButtons.OK, MessageBoxIcon.Information)
    Catch ex As Exception
        MessageBox.Show("Error al subir a Google Drive: " & ex.Message, "Error Crítico", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Finally
        Me.Cursor = Cursors.Default
        btnGuardarDrive.Enabled = True
    End Try
End Sub

Imports Google.Apis.Auth.OAuth2
Imports Google.Apis.Drive.v3
Imports Google.Apis.Services
Imports Google.Apis.Upload
Imports Google.Apis.Util.Store
Imports System.IO
Imports System.Threading

2 referencias
Public Class GoogleDriveHelper

    ' Alcances: Permite para ver y gestionar archivos
    Private Shared ReadOnly SCOPES As String() = {DriveService.Scope.Drive}
    Private Shared ReadOnly APPLICATION_NAME As String = "MarcallaTeX Proformas"

    2 referencias
    Private Shared Function GetService() As DriveService
        Dim credential As UserCredential

        ' Cargar el nuevo JSON de OAuth (client_secret.json)
        Using stream As New FileStream("client_secret.json", FileMode.Open, FileAccess.Read)
            ' Esto abrirá el navegador la primera vez para que inicies sesión
            Dim credPath As String = "token.json"
            credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                GoogleClientSecrets.Load(stream).Secrets,
                SCOPES,
                "user",
                CancellationToken.None,
                New FileDataStore(credPath, True)).Result
        End Using

        ' Crear el servicio de Drive
        Return New DriveService(New BaseClientService.Initializer() With {
            .HttpClientInitializer = credential,
            .ApplicationName = APPLICATION_NAME
        })
    End Function

    ' Obtener o Crear Carpeta (Sin cambios lógicos, solo usa el nuevo servicio)
    1 referencia
    Public Shared Function ObtenerOCrearCarpeta(nombreCarpeta As String, carpetaPadreId As String) As String
        Dim service = GetService()
        nombreCarpeta = nombreCarpeta.Trim()
    End Function
```

```

' Obtener o Crear Carpeta (Sin cambios lógicos, solo usa el nuevo servicio)
1 referencia
Public Shared Function ObtenerOCrearCarpeta(nombreCarpeta As String, carpetaPadreId As String) As String
    Dim service = GetService()
    nombreCarpeta = nombreCarpeta.Trim()

    Dim query As String = $"mimeType='application/vnd.google-apps.folder' and name='{nombreCarpeta}' and '{carpetaPadreId}' in parents and trashed=false"
    Dim request = service.Files.List()
    request.Q = query
    request.Fields = "files(id, name)"

    Dim result = request.Execute()

    If result.Files IsNot Nothing AndAlso result.Files.Count > 0 Then
        Return result.Files(0).Id
    End If

    Dim fileMetadata As New Google.Apis.Drive.v3.Data.File With {
        .Name = nombreCarpeta,
        .MimeType = "application/vnd.google-apps.folder",
        .Parents = New List(Of String) From {carpetaPadreId}
    }

    Dim createRequest = service.Files.Create(fileMetadata)
    createRequest.Fields = "id"

    Dim folder = createRequest.Execute()
    Return folder.Id
End Function

' Subir Archivo (Ahora usa TU cuota de usuario, no fallará)
1 referencia
Public Shared Sub SubirArchivo(rutaArchivo As String, carpetaId As String)
    Dim service = GetService()
    Dim nombreArchivo = Path.GetFileName(rutaArchivo)

    ' Verificar duplicados
    Dim queryDuplicado = $"name='{nombreArchivo}' and '{carpetaId}' in parents and trashed=false"
    Dim reqCheck = service.Files.List()
    reqCheck.Q = queryDuplicado
    Dim resCheck = reqCheck.Execute()

    Dim folder = createRequest.Execute()
    Return folder.Id
End Function

' Subir Archivo (Ahora usa TU cuota de usuario, no fallará)
1 referencia
Public Shared Sub SubirArchivo(rutaArchivo As String, carpetaId As String)
    Dim service = GetService()
    Dim nombreArchivo = Path.GetFileName(rutaArchivo)

    ' Verificar duplicados
    Dim queryDuplicado = $"name='{nombreArchivo}' and '{carpetaId}' in parents and trashed=false"
    Dim reqCheck = service.Files.List()
    reqCheck.Q = queryDuplicado
    Dim resCheck = reqCheck.Execute()

    If resCheck.Files.Count > 0 Then
        ' Borrar versión anterior para actualizar
        service.Files.Delete(resCheck.Files(0).Id).Execute()
    End If

    ' Subir archivo
    Dim fileMetadata As New Google.Apis.Drive.v3.Data.File With {
        .Name = nombreArchivo,
        .Parents = New List(Of String) From {carpetaId}
    }

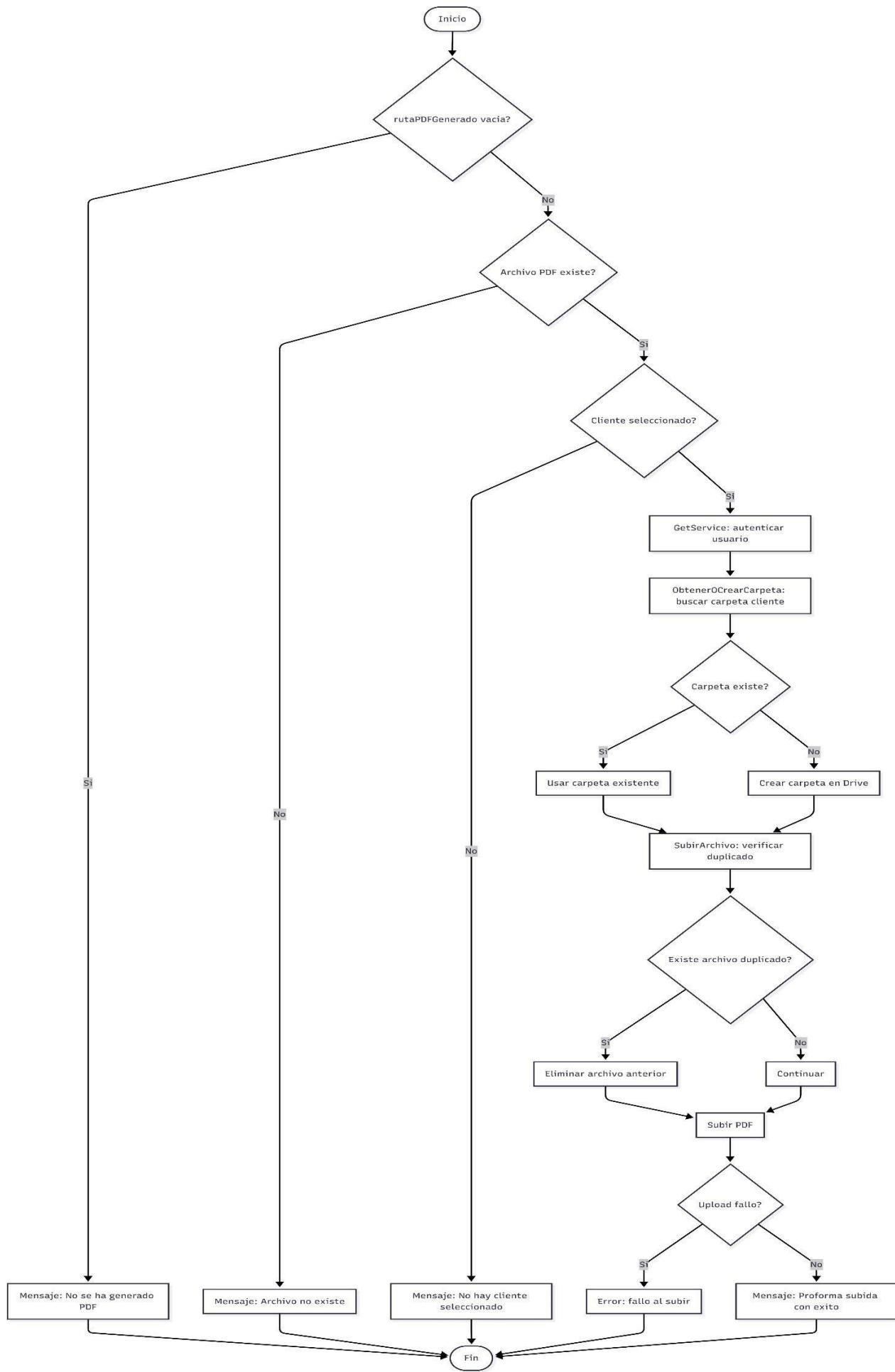
    ' Usamos FileShare.Read para evitar conflictos si el PDF sigue abierto
    Using stream As New FileStream(rutaArchivo, FileMode.Open, FileAccess.Read, FileShare.Read)
        Dim request = service.Files.Create(fileMetadata, stream, "application/pdf")
        request.Fields = "id"

        Dim uploadResult = request.Upload()

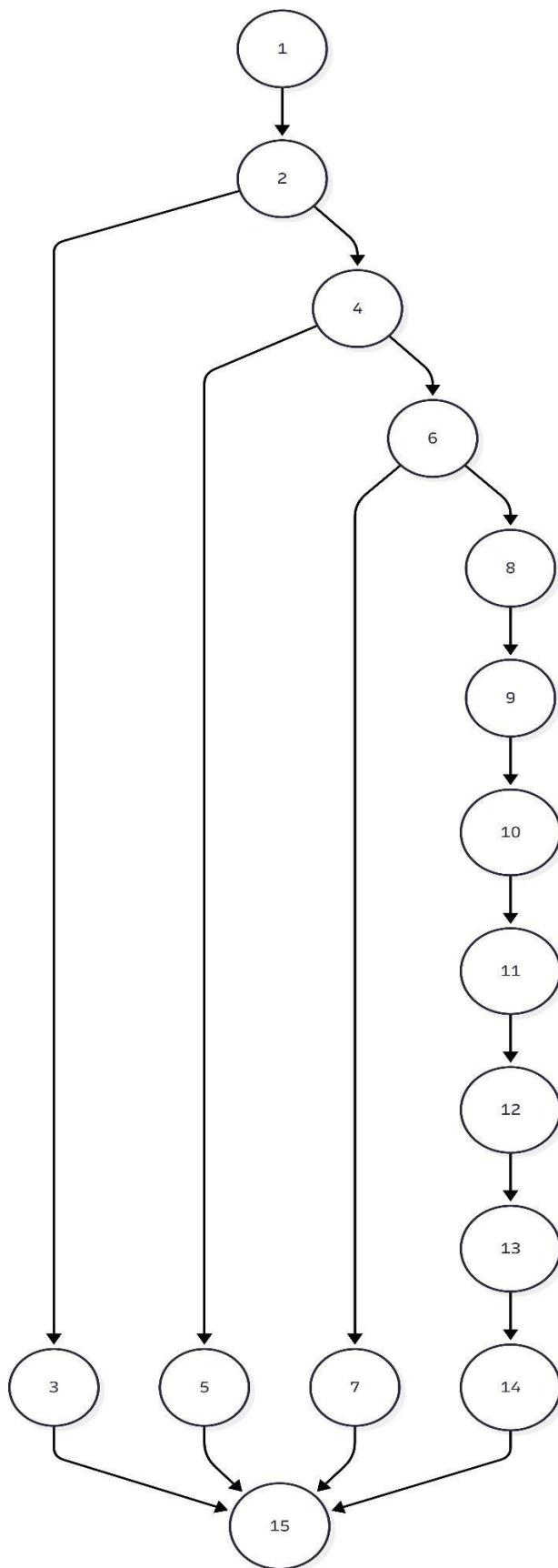
        If uploadResult.Status = UploadStatus.Failed Then
            Throw New Exception("Error de Google Drive: " & uploadResult.Exception.Message)
        End If
    End Using
End Sub
End Class

```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



- 1 = Inicio
- 2 = Validar rutaPDFGenerado
- 3 = Mensaje: falta PDF
- 4 = Validar existencia del archivo
- 5 = Mensaje: archivo no existe
- 6 = Validar cliente seleccionado
- 7 = Mensaje: sin cliente
- 8 = GetService (autenticacion)
- 9 = ObtenerOCrearCarpeta
- 10 = Carpeta existe o se crea
- 11 = SubirArchivo: revisar duplicados
- 12 = Eliminar duplicado (si existe)
- 13 = Subir PDF
- 14 = Mensaje exito / error
- 15 = Fin

4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico)

Ruta	Secuencia de nodos	Descripción
R1	1→2→4→6→8→9→10→11→13→14→15	Flujo correcto: sube proforma a Drive
R2	1→2→3→15	No existe rutaPDFGenerado
R3	1→2→4→5→15	El archivo PDF no existe en disco
R4	1→2→4→6→7→15	No hay cliente seleccionado
R5	1→2→4→6→8→9→10→11→12→13→14→15	Sube reemplazando archivo duplicado
R6	1→2→4→6→8→9→10→11→13→14→15	Sube sin duplicados

5. COMPLEJIDAD CICLOMÁTICA

- N (Número de nodos): 15
- P (Número de nodos predicados): 6
- A (Número de aristas): 20

Cálculo

Fórmula 1:

$$V(G) = P + 1$$

$$V(G) = 6 + 1 = 7$$

Fórmula 2:

$$V(G) = A - N + 2$$

$$V(G) = 20 - 15 + 2 = 7$$

Resultado

La complejidad ciclomática del requisito Historial de Proformas (Google Drive) es 7, lo que indica que existen siete caminos básicos independientes.