Prueba de Caja Blanca

"Generación de proformas MarcallTex"

Integrantes:

Cañola Kevin

Marcalla

Cristhian

Lugamaña

Mateo

Tasiguano

Eduardo

Fecha: 2025/11/25

CONTROL DE VERSIONAMIENTO DE PRUEBAS CB

Versiòn	Fecha	Responsable	Aprobado por
PCB_V1.0.0.docx			

Prueba caja blanca

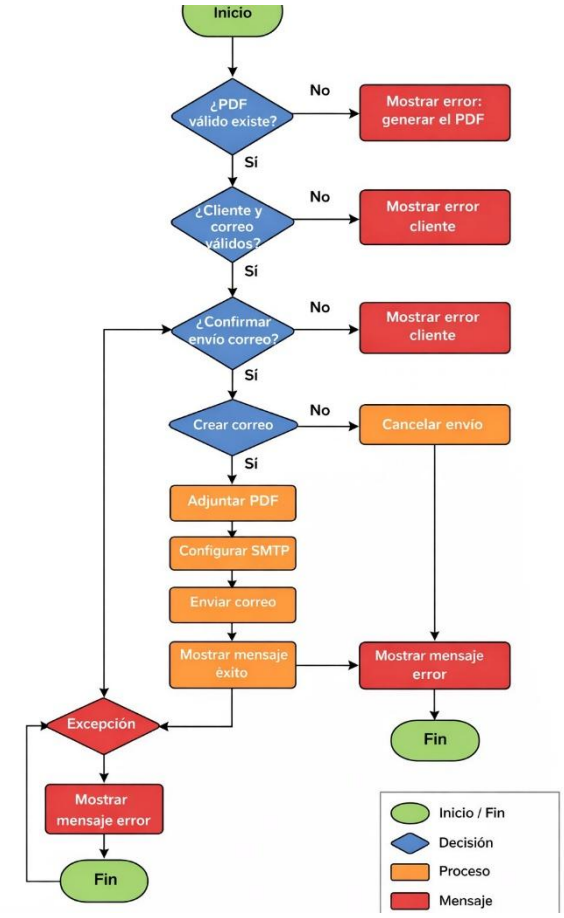
RF N°: REQ012 Envío de Proformas por Correo Electronico

1. **CÓDIGO FUENTE**

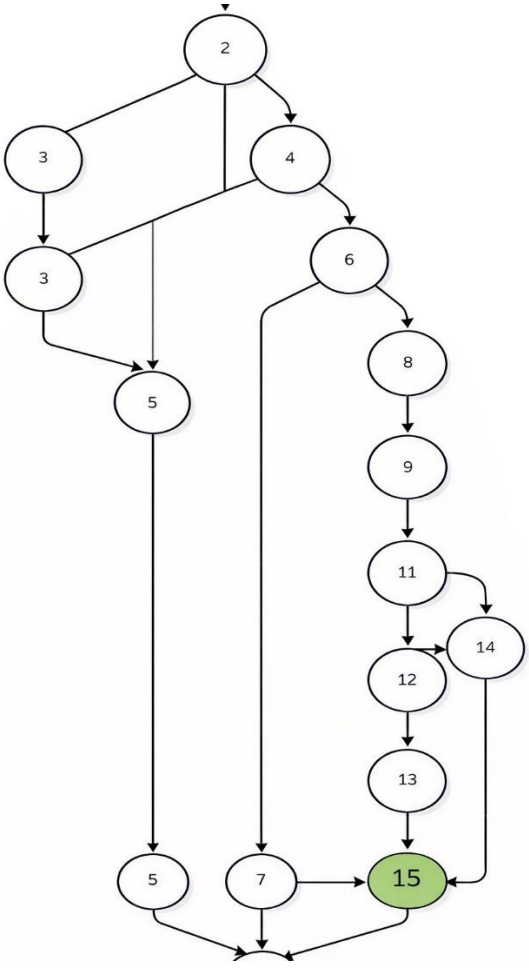
```
Public Sub EnviarCorreoPDF(rutaPDF As String)
    Try
        ' Validar PDF
        If String.IsNullOrEmpty(rutaPDFGenerado) OrElse Not File.Exists(rutaPDFGenerado) Then
            MessageBox.Show("Primero genere el PDF de la proforma.",
                            "Error", MessageBoxButtons.OK,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

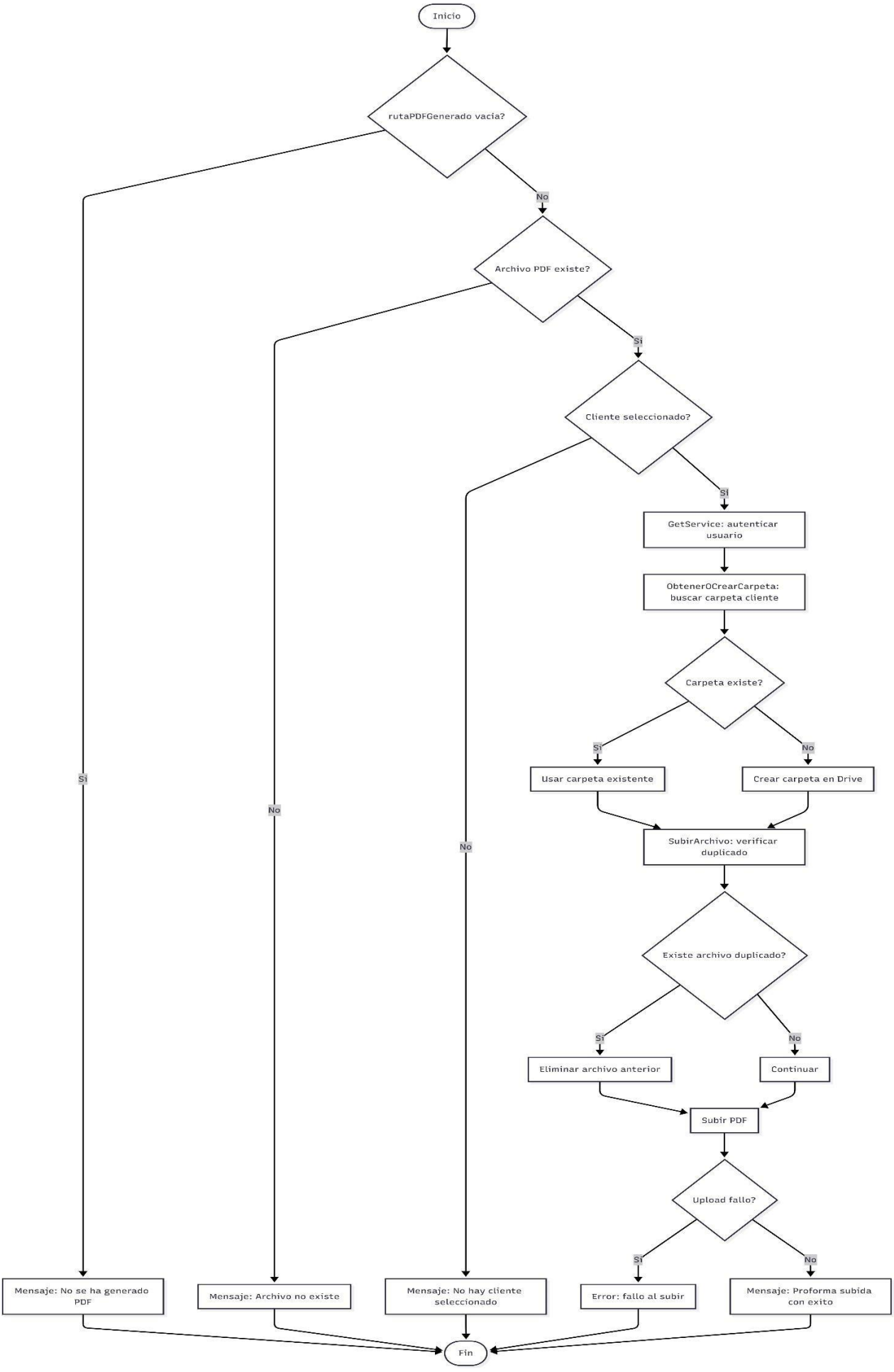
1. CÓDIGO FUENTE

```
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",
        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 = "1EbkQ9taWYrOHBgjVoxsPa8vILsUWS8By"
        Dim nombreClienteFolder As String = clienteSeleccionado.Nombre.Trim().Replace("/", "-").Replace("\", "-")
        Dim carpetaClienteId As String = GoogleDriveHelper.ObtenerOCrearCarpeta(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
```

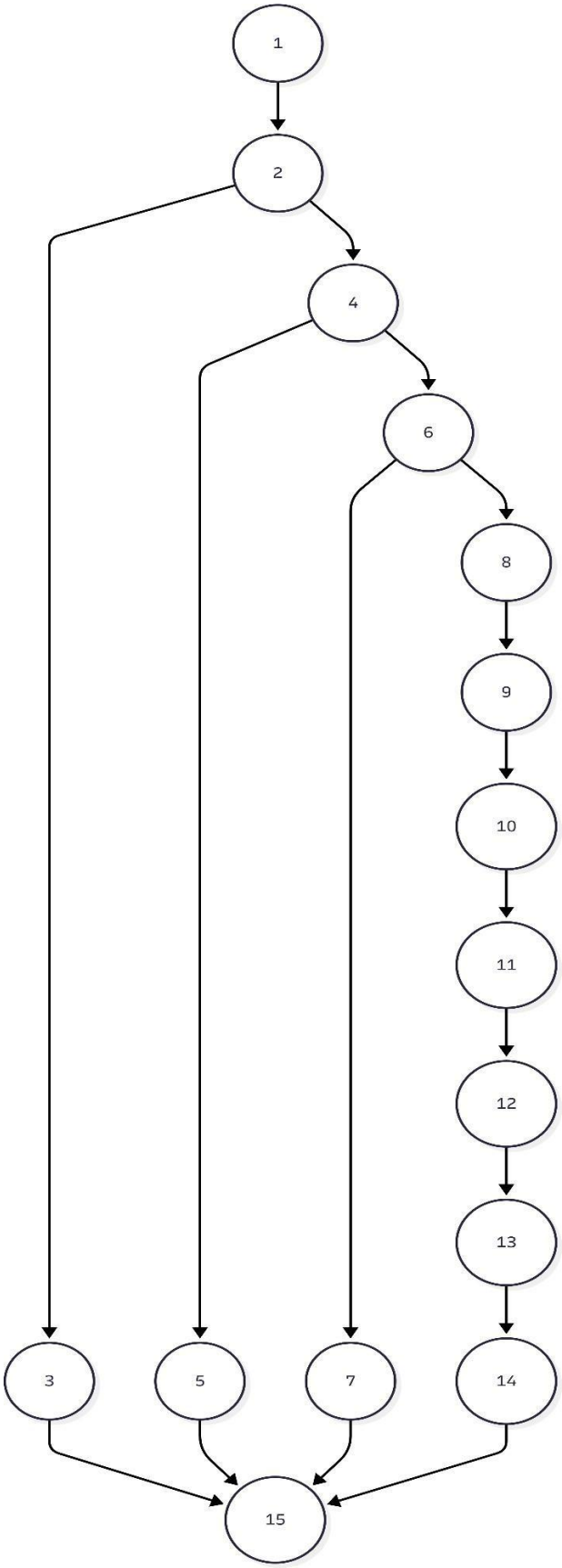
```
Imports Google.Apis.Auth.OAuth2
Imports Google.Apis.Drive.v3
Imports Google.Apis.Services
Imports Google.Apis.Upload
Imports Google.Apis.Util.Store
Imports System.IO
Imports System.Threading

2 referencias
Public Class GoogleDriveHelper

    ' Alcances: Permiso 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()
```

```
' 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.