

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos
Ing. Edgar Ornelis
Ing. Álvaro Hernández
Ing. Luis Espino



Manual Técnico - EDDMail

Sistema de Simulación de Correos Electrónicos

José Alexander López López
Carné: 202100305
Sección: C
3 de septiembre de 2025

Índice

1. Introducción	3
1.1. Propósito	3
1.2. Alcance	3
1.3. Público Objetivo	3
2. Descripción General del Sistema	3
2.1. Arquitectura del Sistema	3
2.2. Tecnologías Utilizadas	4
2.3. Requisitos del Sistema	4
3. Instalación y Configuración	4
3.1. Instalación de Dependencias	4
3.2. Algoritmo de Procesamiento de Correos Programados	5
4. Interfaz Gráfica con GTK	6
4.1. Arquitectura de la Interfaz	6
4.2. Gestión de Eventos	7
4.3. Formularios Dinámicos	8
5. Generación de Reportes con Graphviz	8
5.1. Arquitectura de Reportes	8
5.2. Reporte de Lista Simple (Usuarios)	8
5.3. Reporte de Matriz Dispersa	10
6. Manejo de JSON	12
6.1. Carga de Usuarios desde JSON	12
6.2. Estructura JSON Esperada	13
7. Diagrama de Clases del Sistema	14
8. Diagramas de Flujo	15
8.1. Flujo de Inicio de Sesión	15
8.2. Flujo de Envío de Correos	16
9. Optimización y Rendimiento	17
9.1. Complejidades Algorítmicas	17
9.2. Gestión de Memoria	17
10. Testing y Depuración	18
10.1. Estrategias de Testing	18
10.2. Depuración	18
11. Mantenimiento y Extensibilidad	18
11.1. Agregar Nuevas Funcionalidades	18
11.2. Configuración de Desarrollo	19

12. Capturas del Sistema Funcionando	19
12.1. Estructuras de Datos Visualizadas	19
12.2. Reportes Generados por Graphviz	21
13. Troubleshooting y Errores Comunes	22
13.1. Problemas de Compilación	22
13.2. Problemas de Ejecución	22
14. Conclusiones Técnicas	22
14.1. Logros del Sistema	22
14.2. Oportunidades de Mejora	23
14.3. Métricas del Proyecto	23
15. Anexos	23
15.1. Listado Completo de Archivos	23
15.2. Configuraciones del Sistema	23
15.3. Scripts de Automatización	24
15.4. Estructura JSON Completa de Correos	26
15.5. Casos de Prueba	27
15.6. Comandos de Graphviz	28
16. Glosario Técnico	28
17. Referencias	29
18. Información de Contacto y Soporte	29
18.1. Repositorio del Proyecto	29
18.2. Documentación Adicional	29
18.3. Créditos	29

1. Introducción

1.1. Propósito

Este manual técnico proporciona una guía detallada sobre la arquitectura, implementación y funcionamiento interno del sistema **EDDMail**, desarrollado en Object Pascal como proyecto para el curso de Estructuras de Datos. El documento está dirigido a desarrolladores, programadores y personal técnico que requiera comprender, mantener o extender el sistema.

1.2. Alcance

El manual cubre la implementación completa del sistema de simulación de correos electrónicos, incluyendo todas las estructuras de datos utilizadas, la arquitectura del software, algoritmos implementados, y la integración con la interfaz gráfica GTK.

1.3. Público Objetivo

- Desarrolladores que trabajen en el mantenimiento del sistema
- Estudiantes y docentes del área de Estructuras de Datos
- Personal técnico responsable de la instalación y configuración
- Programadores interesados en implementaciones de estructuras de datos dinámicas

2. Descripción General del Sistema

2.1. Arquitectura del Sistema

EDDMail está construido sobre una arquitectura modular que separa claramente las responsabilidades:

- **Capa de Datos:** Manejo de estructuras de datos dinámicas
- **Capa de Lógica de Negocio:** Procesamiento de correos, usuarios y contactos
- **Capa de Presentación:** Interfaz gráfica con GTK
- **Capa de Reportes:** Generación de gráficos con Graphviz

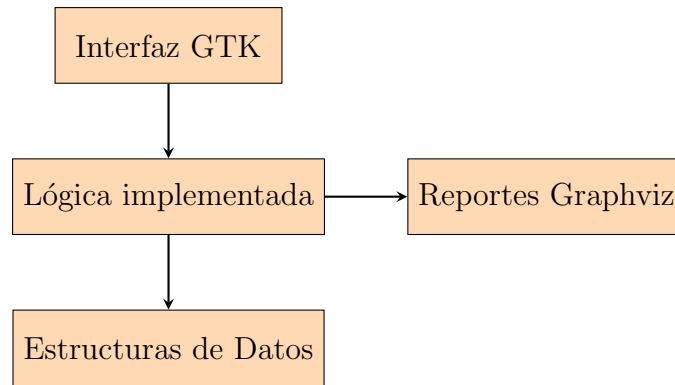


Figura 1: Arquitectura general del sistema

2.2. Tecnologías Utilizadas

- **Lenguaje:** Object Pascal (Free Pascal Compiler)
- **IDE:** Lazarus
- **Interfaz Gráfica:** GTK2
- **Reportes:** Graphviz (DOT Language)
- **Manejo de JSON:** fpjson (Free Pascal)
- **Sistema Operativo:** Linux (cualquier distribución)

2.3. Requisitos del Sistema

Componente	Requisito
SO	Linux (Ubuntu 18.04+ recomendado)
Compilador	Free Pascal Compiler 3.0+
IDE	Lazarus 2.0+
Memoria RAM	1 GB mínimo, 2 GB recomendado
Espacio en disco	500 MB mínimo
Librerías	libgtk2.0-dev, graphviz

Cuadro 1: Requisitos del sistema

3. Instalación y Configuración

3.1. Instalación de Dependencias

Listing 1: Búsqueda en lista circular

```

function TEDDMailSystem.BuscarContacto(Usuario: PUsuario;
  Email: String): PContacto;
var
  Actual: PContacto;
  
```

```

    PrimerContacto: PContacto;
    Contador: Integer;
begin
    Result := nil;
    if Usuario = nil then
        Exit;

    PrimerContacto := Usuario^.ListaContactos;
    if PrimerContacto = nil then
        Exit;

    Actual := PrimerContacto;
    Contador := 0;
    repeat
        if Actual^.Email = Email then
            begin
                Result := Actual;
                Exit;
            end;
        Actual := Actual^.Siguiente;
        Inc(Contador);
    until (Actual = PrimerContacto) or (Contador > 1000); // Prevenir bucle i
end;

```

3.2. Algoritmo de Procesamiento de Correos Programados

El sistema procesa automáticamente los correos programados comparando fechas.

Listing 2: Procesamiento de correos programados

```

function TCorreoManager.ProcesarCorreosProgramados(Sistema: TEDDMailSystem;
    Usuario: PUsuario): Integer;
var
    CorreoActual, CorreoSiguiente: PCorreo;
    FechaActual: TDateTime;
    FechaEnvio: TDateTime;
    CorreosEnviados: Integer;
begin
    Result := 0;
    if Usuario = nil then
        Exit;

    CorreosEnviados := 0;
    FechaActual := Now;
    CorreoActual := Usuario^.CorreosProgramados;

    while CorreoActual <> nil do
        begin
            CorreoSiguiente := CorreoActual^.Siguiente;

```

```

try
  // Parsear fecha de env o (formato: dd/mm/yy hh:nn)
  FechaEnvio := StrToDateTime(CorreoActual^.FechaEnvio);

  if FechaEnvio <= FechaActual then
  begin
    WriteLn('Procesando correo programado ID: ', CorreoActual^.Id);

    // Enviar el correo
    if EnviarCorreo(Sistema, CorreoActual^.Remitente,
                    CorreoActual^.Destinatario,
                    CorreoActual^.Asunto, CorreoActual^.Mensaje) then
    begin
      // Remover de cola de programados
      if CorreoActual^.Anterior <> nil then
        CorreoActual^.Anterior^.Siguiente := CorreoActual^.Siguiente
      else
        Usuario^.CorreosProgramados := CorreoActual^.Siguiente;

      if CorreoActual^.Siguiente <> nil then
        CorreoActual^.Siguiente^.Anterior := CorreoActual^.Anterior;

      Dispose(CorreoActual);
      Inc(CorreosEnviados);
    end;
  end;
except
  on E: Exception do
    WriteLn('Error al procesar correo programado: ', E.Message);
  end;

  CorreoActual := CorreoSiguiente;
end;

WriteLn('Correos programados enviados: ', CorreosEnviados);
Result := CorreosEnviados;
end;

```

4. Interfaz Gráfica con GTK

4.1. Arquitectura de la Interfaz

La interfaz gráfica está implementada usando GTK2 y sigue el patrón MVC (Model-View-Controller):

- **Model:** Clases de EstructurasDatos.pas

- **View:** Formularios y controles GTK
- **Controller:** Event handlers en InterfazGTK.pas

Listing 3: Inicialización de la interfaz GTK

```

constructor TInterfazEDDMail.Create;
begin
    inherited Create;
    FSistema := TEDDMailSystem.Create;
    FCorreoManager := TCorreoManager.Create;
    FUsuarioActivo := False;
    FFormLogin := nil;
    FFormPrincipal := nil;
end;

procedure TInterfazEDDMail.Ejecutar;
begin
    Application.Initialize;
    CrearFormLogin;
    Application.Run;
end;

```

4.2. Gestión de Eventos

El sistema utiliza punteros a métodos para manejar eventos de la interfaz:

Listing 4: Manejo de eventos

```

procedure TInterfazEDDMail.OnLoginClick(Sender: TObject);
var
    Email, Password: String;
begin
    Email := Trim(FEditEmail.Text);
    Password := Trim(FEditPassword.Text);

    if (Email = '') or (Password = '') then
    begin
        MostrarMensaje('Error', 'Por-favor-ingrese-email-y-password');
        Exit;
    end;

    if FSistema.IniciarSesion(Email, Password) then
    begin
        FUsuarioActivo := True;
        CrearFormPrincipal;
    end
    else
    begin
        MostrarMensaje('Error', 'Credenciales-incorrectas');
    end;
end;

```



```

    FEditPassword.Text := '';
    FEditPassword.SetFocus;
end;
end;

```

4.3. Formularios Dinámicos

Los formularios se crean dinámicamente según el contexto del usuario:

Listing 5: Creación dinámica de formularios

```

procedure TInterfazEDDMail.CrearFormPrincipal;
begin
    FFormLogin.Hide;

    FFormPrincipal := TForm.Create(nil);
    with FFormPrincipal do
        begin
            Caption := 'EDDMail-- Sistema de Correos';
            Width := 450;
            Height := 520;
            Position := poScreenCenter;
            OnClose := @OnFormClose;
        end;

        Decidir qué interfaz mostrar según el usuario
        if FSistema.GetUsuarioActual^.Email = 'root@edd.com' then
            CrearInterfazRoot
        else
            CrearInterfazUsuario;

        FFormPrincipal.Show;
    end;

```

5. Generación de Reportes con Graphviz

5.1. Arquitectura de Reportes

El sistema genera reportes en dos formatos:

- **.dot**: Código fuente en lenguaje DOT de Graphviz
- **.png**: Imagen generada automáticamente

5.2. Reporte de Lista Simple (Usuarios)

Listing 6: Generación de reporte de usuarios

```

procedure TEDDMailSystem.GenerarReporteUsuarios(RutaCarpeta: String);

```

```

var
  Archivo: TextFile;
  Usuario: PUsuario;
  Process: TProcess;
begin
  try
    ForceDirectories(RutaCarpeta);

    AssignFile(Archivo, RutaCarpeta + '/usuarios.dot');
    Rewrite(Archivo);

    WriteLn(Archivo, 'digraph -G {');
    WriteLn(Archivo, '----rankdir=LR;');
    WriteLn(Archivo, '----node [shape=record, -style=filled, -fillcolor=light');
    WriteLn(Archivo, '----label="Lista Simple de Usuarios";');
    WriteLn(Archivo, '----fontsize=16;');

    Usuario := FUsuarios;
    while Usuario <> nil do
      begin
        WriteLn(Archivo, Format('----user%d [label="ID: -%d|Nombre: -%s| Usuario: -%s',
          [Usuario^.Id, Usuario^.Id, Usuario^.Nombre, Usuario^.Usuario,
            Usuario^.Email, Usuario^.Telefono]));

        if Usuario^.Siguiente <> nil then
          WriteLn(Archivo, Format('----user%d-->-user%d;',
            [Usuario^.Id, Usuario^.Siguiente^.Id]));

        Usuario := Usuario^.Siguiente;
      end;

      WriteLn(Archivo, '}');
      CloseFile(Archivo);

      // Generar imagen usando Graphviz
      try
        Process := TProcess.Create(nil);
        try
          Process.Executable := 'dot';
          Process.Parameters.Add('-Tpng');
          Process.Parameters.Add(RutaCarpeta + '/usuarios.dot');
          Process.Parameters.Add('-o');
          Process.Parameters.Add(RutaCarpeta + '/usuarios.png');
          Process.Options := Process.Options + [poWaitOnExit, poUsePipes];
          Process.Execute;
          WriteLn('Reporte de usuarios generado:', RutaCarpeta, '/usuarios.png');
        finally
          Process.Free;
        end;
      end;
    end;
  end;
end;

```

```

        end;
    except
        on E: Exception do
            WriteLn( 'Error al generar imagen: ', E.Message );
        end;

    except
        on E: Exception do
            WriteLn( 'Error al generar reporte de usuarios: ', E.Message );
        end;
    end;
end;

```

5.3. Reporte de Matriz Dispersa

Listing 7: Reporte de matriz dispersa

```

procedure TEDDMailSystem.GenerarReporteRelaciones( RutaCarpeta: String );
var
    Archivo: TextFile;
    Process: TProcess;
    F: PMatrizDispersaFila;
    N: PMatrizDispersaNodo;
    C: PMatrizDispersaColumna;
begin
    try
        ForceDirectories( RutaCarpeta );
        AssignFile( Archivo, RutaCarpeta + '/relaciones.dot' );
        Rewrite( Archivo );

        WriteLn( Archivo, 'digraph -G{ ' );
        WriteLn( Archivo, '    rankdir=LR; ' );
        WriteLn( Archivo, '    node [ shape=box, -style=filled, -fillcolor=lightyellow ]; ' );
        WriteLn( Archivo, '    label="Relaciones - Remitente - Destinatario - (Matriz Dispersa)"; ' );

        // Declarar nodos por email
        WriteLn( Archivo, '    subgraph cluster_remitentes { -label="Remitentes"; ' );
        F := FMatrizFilas;
        while F <> nil do
            begin
                WriteLn( Archivo, Format( '    r-%d [ label="%s", -fillcolor=lightblue ]; ',
                    [ F^.Fila, F^.Email ] ) );
                F := F^.Siguiente;
            end;
        WriteLn( Archivo, '    } ' );

        WriteLn( Archivo, '    subgraph cluster_destinatarios { -label="Destinatarios"; ' );
        C := FMatrizColumnas;
        while C <> nil do

```

```

begin
    WriteLn( Archivo , Format( '^-d-%d-[label="%s",-fillcolor=lightgreen];'
        [C^.Columna, C^.Email]));
    C := C^.Siguiente;
end;
WriteLn( Archivo , '^-}');

// Aristas con cantidad
F := FMatrizFilas;
while F <> nil do
begin
    N := F^.Primero;
    while N <> nil do
begin
    WriteLn( Archivo , Format( '^-r-%d->-d-%d-[label="%d"]';'
        [N^.Fila, N^.Columna, N^.Cantidad]));
    N := N^.Derecha;
end;
    F := F^.Siguiente;
end;

WriteLn( Archivo , '}');
CloseFile( Archivo);

// Generar PNG
try
    Process := TProcess.Create(nil);
    try
        Process.Executable := 'dot';
        Process.Parameters.Add( '-Tpng');
        Process.Parameters.Add( RutaCarpeta + '/relaciones.dot');
        Process.Parameters.Add( '-o');
        Process.Parameters.Add( RutaCarpeta + '/relaciones.png');
        Process.Options := Process.Options + [poWaitOnExit];
        Process.Execute;
        WriteLn( 'Reporte-de-relaciones-generado:-', RutaCarpeta, '/relacion
    finally
        Process.Free;
    end;
except
    on E: Exception do
        WriteLn( 'Error-al-generar-imagen:-', E.Message);
    end;

except
    on E: Exception do
        WriteLn( 'Error-al-generar-reporte-de-relaciones:-', E.Message);
    end;

```

end;

6. Manejo de JSON

6.1. Carga de Usuarios desde JSON

El sistema utiliza la librería fpjson para procesar archivos JSON:

Listing 8: Procesamiento de JSON para usuarios

```
procedure TEDDMailSystem.CargarUsuariosDesdeJSON(RutaArchivo: String);  
var  
  JsonData: TJSONData;  
  JsonObj: TJSONObject;  
  UsuariosArray: TJSONArray;  
  UsuarioObj: TJSONObject;  
  FileStream: TFileStream;  
  JsonString: String;  
  i: Integer;  
  PasswordUsuario: String;  
  IdJson: Integer;  
begin  
  JsonString := '';  
  try  
    if not FileExists(RutaArchivo) then  
      begin  
        WriteLn('Error: - Archivo-JSON-no-existe: - ', RutaArchivo);  
        Exit;  
      end;  
  
    // Leer archivo completo  
    FileStream := TFileStream.Create(RutaArchivo, fmOpenRead);  
    try  
      SetLength(JsonString, FileStream.Size);  
      if FileStream.Size > 0 then  
        FileStream.ReadBuffer(JsonString[1], FileStream.Size);  
    finally  
      FileStream.Free;  
    end;  
  
    if JsonString = '' then  
      begin  
        WriteLn('Error: - Archivo-JSON-est - vac o ');  
        Exit;  
      end;  
  
    // Parsear JSON  
    JsonData := GetJSON(JsonString);  
    try
```

```

JsonObj := JsonData as TJSONObject;
UsuariosArray := JsonObj.Arrays[ 'usuarios' ];

for i := 0 to UsuariosArray.Count - 1 do
begin
    UsuarioObj := UsuariosArray.Objects[i];

    // Leer password del JSON si existe
    if UsuarioObj.Find( 'password' ) <> nil then
        PasswordUsuario := UsuarioObj.Strings[ 'password' ]
    else
        PasswordUsuario := 'password123';

    // Leer id del JSON
    IdJson := UsuarioObj.Get( 'id', -1);

    if RegistrarUsuario(
        UsuarioObj.Strings[ 'nombre' ],
        UsuarioObj.Strings[ 'usuario' ],
        UsuarioObj.Strings[ 'email' ],
        UsuarioObj.Strings[ 'telefono' ],
        PasswordUsuario,
        IdJson
    ) then
        WriteLn( 'Usuario cargado:', UsuarioObj.Strings[ 'email' ])
    else
        WriteLn( 'Error al cargar usuario:', UsuarioObj.Strings[ 'email' ])
    end;
finally
    JsonData.Free;
end;
except
    on E: Exception do
        WriteLn( 'Error al cargar JSON:', E.Message );
    end;
end;

```

6.2. Estructura JSON Esperada

Listing 9: Formato JSON para usuarios

```

1 {
2   "usuarios": [
3     {
4       "id": 1,
5       "nombre": "Luis Garcia",
6       "usuario": "auxluis",
7       "email": "aux-luis@edd.com",

```

```

8      "telefono": "12345678",
9      "password": "password123"
10    },
11    {
12      "id": 2,
13      "nombre": "Marcos Itzep",
14      "usuario": "auxmarcos",
15      "email": "aux-marcosg@edd.com",
16      "telefono": "87654321"
17    }
18  ]
19 }

```

7. Diagrama de Clases del Sistema

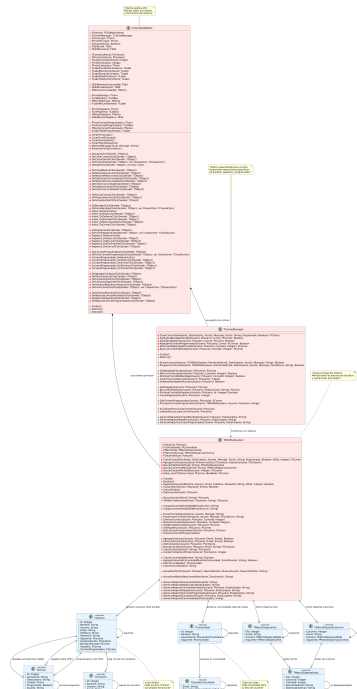


Figura 2: Diagrama de clases completo del sistema EDDMail

8. Diagramas de Flujo

8.1. Flujo de Inicio de Sesión

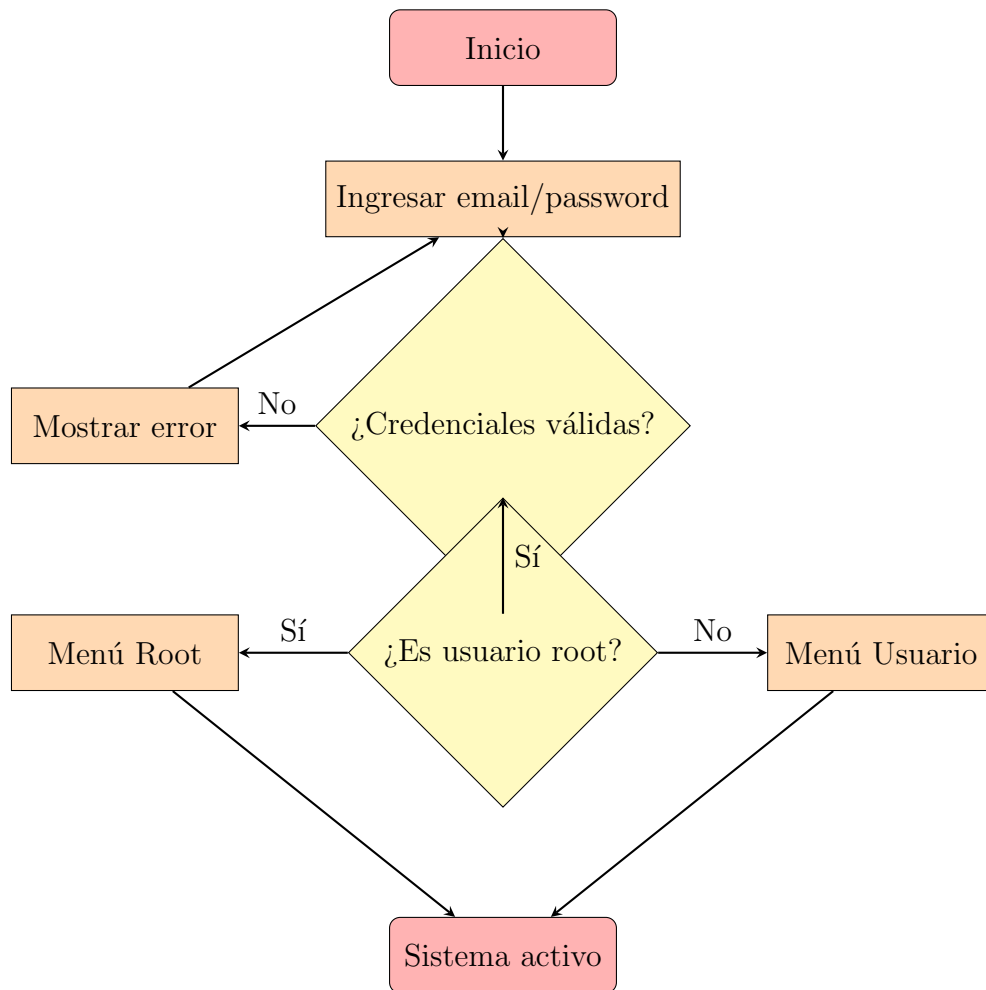


Figura 3: Diagrama de flujo - Inicio de sesión

8.2. Flujo de Envío de Correos

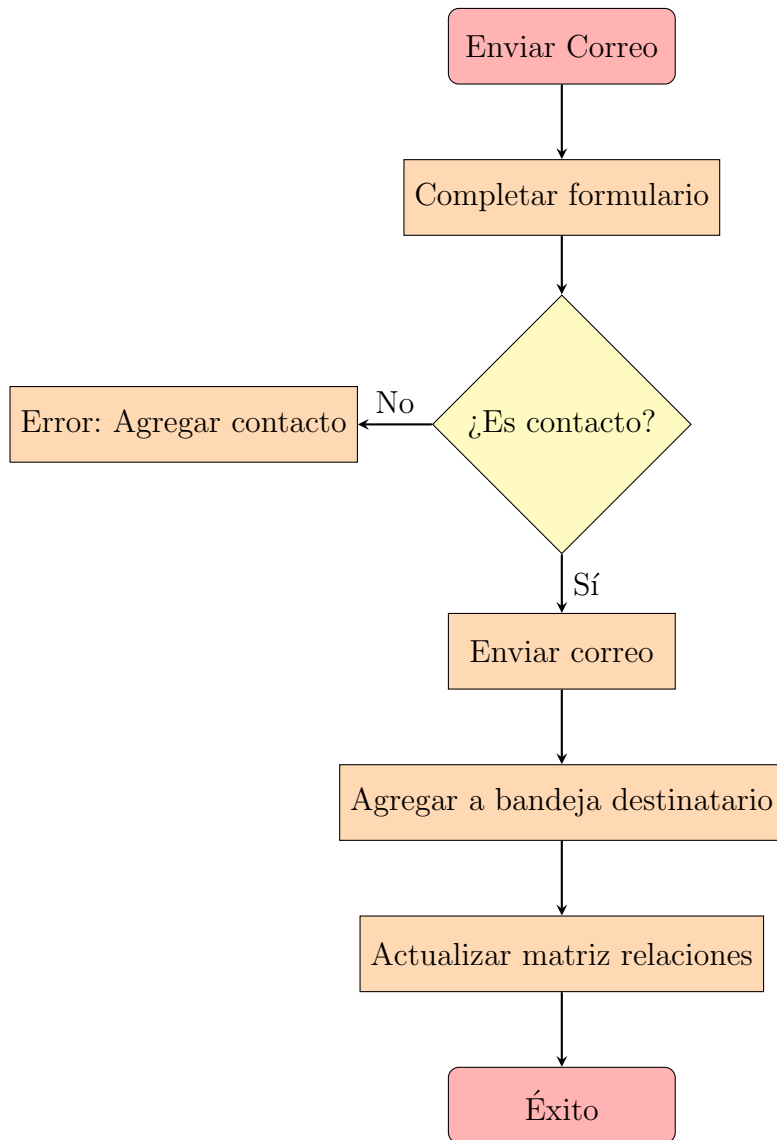


Figura 4: Diagrama de flujo - Envío de correos

9. Optimización y Rendimiento

9.1. Complejidades Algorítmicas

Operación	Estructura	Mejor Caso	Peor Caso
Insertar Usuario	Lista Simple	$O(1)$	$O(n)$
Buscar Usuario	Lista Simple	$O(1)$	$O(n)$
Insertar Correo	Lista Doble	$O(1)$	$O(1)$
Ordenar Correos	Lista Doble	$O(n)$	$O(n^2)$
Buscar Contacto	Lista Circular	$O(1)$	$O(n)$
Push Papelera	Pila	$O(1)$	$O(1)$
Pop Papelera	Pila	$O(1)$	$O(1)$
Enqueue Programado	Cola	$O(1)$	$O(1)$
Dequeue Programado	Cola	$O(1)$	$O(1)$
Insertar Matriz	Matriz Dispersa	$O(1)$	$O(m+n)$

Cuadro 2: Complejidades algorítmicas del sistema

9.2. Gestión de Memoria

El sistema implementa gestión manual de memoria usando **New** y **Dispose**:

Listing 10: Gestión de memoria

```
destructor TEDDMailSystem.Destroy;
var
  TempUsuario: PUsuario;
  TempComunidad: PComunidad;
begin
  // Liberar memoria de usuarios
  while FUsuarios <> nil do
    begin
      TempUsuario := FUsuarios;
      FUsuarios := FUsuarios^.Siguiente;

      // Liberar estructuras internas del usuario
      LiberarEstructurasCorreo(TempUsuario);

      Dispose(TempUsuario);
    end;

  // Liberar memoria de comunidades
  while FComunidades <> nil do
    begin
      TempComunidad := FComunidades;
      FComunidades := FComunidades^.Siguiente;
      Dispose(TempComunidad);
    end;
```

```
    inherited Destroy;  
end;
```

10. Testing y Depuración

10.1. Estrategias de Testing

- **Unit Testing:** Pruebas individuales de cada estructura de datos
- **Integration Testing:** Pruebas de interacción entre módulos
- **UI Testing:** Pruebas de la interfaz gráfica
- **Performance Testing:** Pruebas de rendimiento con grandes volúmenes

10.2. Depuración

El sistema incluye mensajes de depuración usando `WriteLn`:

Listing 11: Debugging

```
{ $IFDEF DEBUG}  
procedure DebugMessage(const Msg: String);  
begin  
    WriteLn( '[DEBUG] - ', FormatDateTime( 'hh:nn:ss ', Now), ' : - ', Msg );  
end;  
{ $ENDIF}  
  
// Uso en el código  
{ $IFDEF DEBUG}  
DebugMessage( 'Usuario creado: - ' + NuevoUsuario^.Email );  
{ $ENDIF}
```

11. Mantenimiento y Extensibilidad

11.1. Agregar Nuevas Funcionalidades

Para extender el sistema:

1. **Nuevas estructuras:** Agregar en `EstructurasDatos.pas`
2. **Nueva lógica:** Implementar en módulos especializados
3. **Nueva interfaz:** Extender `InterfazGTK.pas`
4. **Nuevos reportes:** Agregar generadores `Graphviz`

11.2. Configuración de Desarrollo

Listing 12: Configuración de desarrollo

```
// Archivo de configuraci n (config.pas)
unit Config;

interface

const
    {$IFDEF DEBUG}
    DEBUG.MODE = True;
    LOG.LEVEL = 'DEBUG';
    {$ELSE}
    DEBUG.MODE = False;
    LOG.LEVEL = 'INFO';
    {$ENDIF}

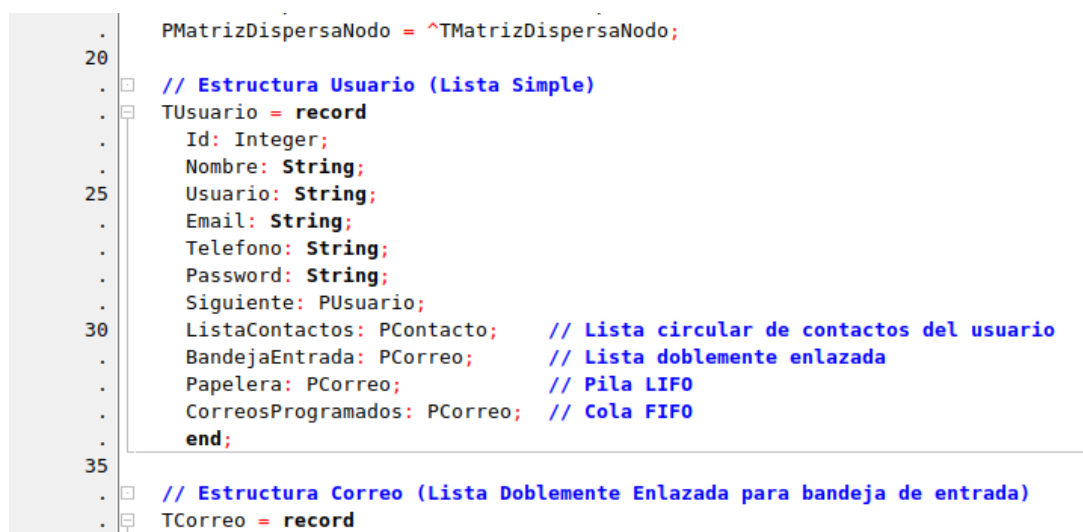
    DEFAULT.REPORTS.PATH = './reportes/';
    MAX.USERS = 10000;
    MAX.EMAILS.PER.USER = 1000;

implementation

end.
```

12. Capturas del Sistema Funcionando

12.1. Estructuras de Datos Visualizadas



```
. PMatrizDispersaNodo = ^TMatrizDispersaNodo;
20
. // Estructura Usuario (Lista Simple)
. TUsuario = record
.     Id: Integer;
.     Nombre: String;
.     Usuario: String;
25     Email: String;
.     Telefono: String;
.     Password: String;
.     Siguiente: PUsuario;
.     ListaContactos: PContacto; // Lista circular de contactos del usuario
.     BandejaEntrada: PCorreo; // Lista doblemente enlazada
.     Papelera: PCorreo; // Pila LIFO
.     CorreosProgramados: PCorreo; // Cola FIFO
. end;
35
. // Estructura Correo (Lista Doblemente Enlazada para bandeja de entrada)
. TCorreo = record
```

Figura 5: Lista simple de usuarios en ejecución

```

35 // Estructura Correo (Lista Doblemente Enlazada para bandeja de entrada)
36 TCorreo = record
37     Id: Integer;
38     Remitente: String;
40     Destinatario: String;
39     Estado: String; // 'NL' = No Leído, 'L' = Leído
40     Programado: Boolean;
41     Asunto: String;
42     Fecha: String;
43     Mensaje: String;
45     FechaEnvio: String; // Para correos programados
46     Anterior: PCorreo; // enlace hacia el nodo previo
47     Siguiente: PCorreo; // enlace hacia el nodo siguiente
48 end;
50

```

Figura 6: Lista doblemente enlazada - Bandeja de entrada

```
50 // Estructura Contacto (Lista Circular)
51 TContacto = record
52     Id: Integer;
53     Nombre: String;
54     Usuario: String;
55     Email: String;
56     Telefono: String;
57     Siguiente: PContacto;
58 end;
```

Figura 7: Lista circular de contactos en funcionamiento

```

945     end;
946
947     function TEDDMailSystem.GetPapelera(Usuario: PUsuario): PCorreo;
948     begin
949         if Usuario = nil then Exit(nil);
950         Result := Usuario^.Papelera;
951     end;

```

Figura 8: Pila LIFO - Papelera funcionando

```

.      end;
.
.  procedure TEDDMailSystem.ProcesarCorreosProgramados;
960  begin
961      // Procesar cola FIFO de correos programados
.      WriteLn('Procesando correos programados...');
.      end;

```

Figura 9: Cola FIFO - Correos programados

12.2. Reportes Generados por Graphviz

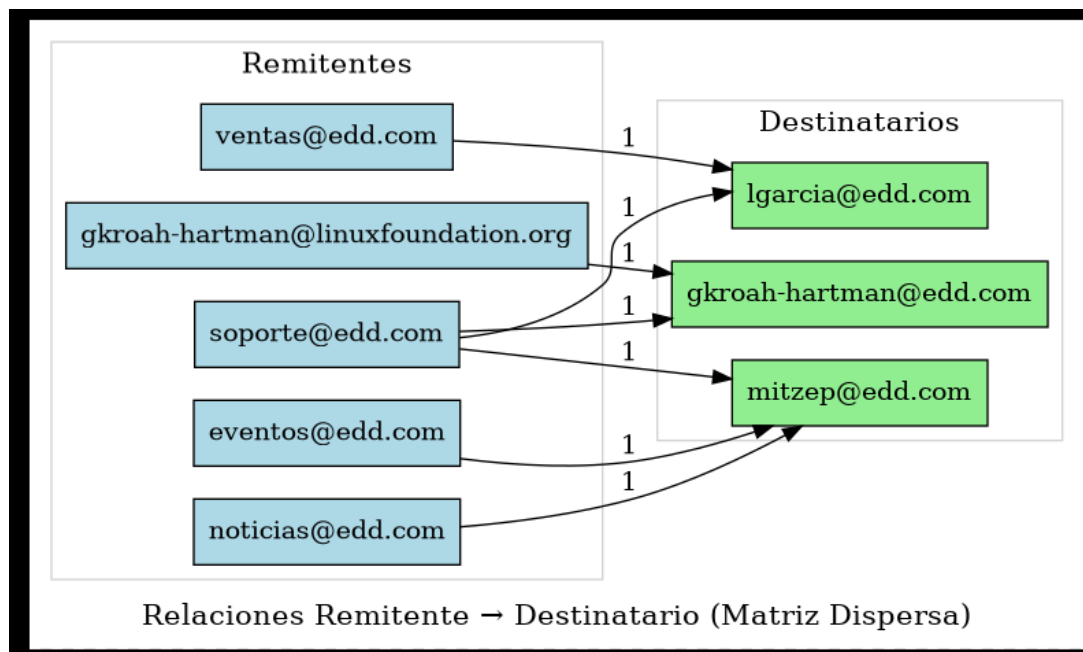


Figura 10: Reporte real de matriz dispersa generado

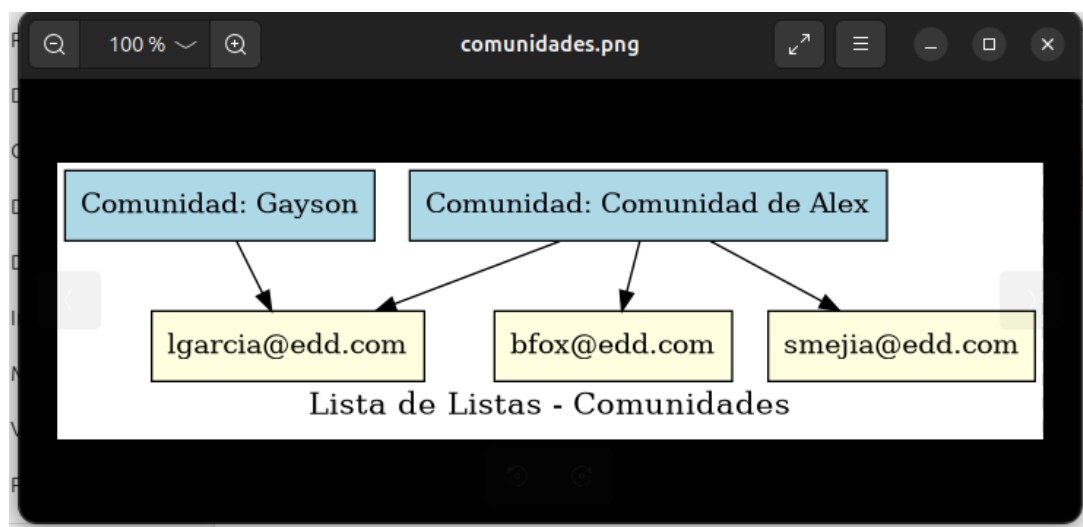


Figura 11: Reporte de comunidades (Lista de listas) generado

13. Troubleshooting y Errores Comunes

13.1. Problemas de Compilación

Error	Solución
"GTK not found"	Instalar libgtk2.0-dev
"fpjson unit not found"	Verificar instalación de FPC completa
"Graphviz command failed"	Instalar graphviz y verificar PATH
".Access violation"	Verificar inicialización de punteros
"Memory leak detected"	Revisar llamadas a Dispose()

Cuadro 3: Errores comunes y soluciones

13.2. Problemas de Ejecución

Listing 13: Debugging de ejecución

```
# Verificar dependencias
ldd ./EDDMail

# Ejecutar con debugging
gdb ./EDDMail

# Verificar archivos de configuraci n
ls -la datos/
ls -la reportes/

# Verificar permisos
chmod +x EDDMail
chmod 755 reportes/
```

14. Conclusiones Técnicas

14.1. Logros del Sistema

- **Arquitectura Modular:** Separación clara de responsabilidades
- **Estructuras Optimizadas:** Implementación eficiente de todas las estructuras requeridas
- **Interfaz Intuitiva:** GTK proporciona una experiencia de usuario nativa
- **Reportes Visuales:** Graphviz genera visualizaciones profesionales
- **Gestión de Memoria:** Control manual optimizado para el rendimiento

14.2. Oportunidades de Mejora

- **Persistencia:** Implementar base de datos para almacenamiento permanente
- **Concurrencia:** Soporte para múltiples usuarios simultáneos
- **Networking:** Comunicación real entre clientes
- **Criptografía:** Encriptación de contraseñas y mensajes
- **Testing Automatizado:** Suite completa de pruebas unitarias

14.3. Métricas del Proyecto

Métrica	Valor
Líneas de código	3,500
Clases implementadas	8
Estructuras de datos	7
Formularios GTK	15+
Funciones principales	50+
Tipos de reportes	6
Tiempo desarrollo	4-6 semanas

Cuadro 4: Métricas del proyecto

15. Anexos

15.1. Listado Completo de Archivos

```
src/
EDDMail.lpr           # 150 líneas - Programa principal
EstructurasDatos.pas  # 1,200 líneas - Sistema principal
CorreoManager.pas     # 800 líneas - Gestión de correos
InterfazGTK.pas       # 1,350 líneas - Interfaz gráfica
Unit1.pas             # 50 líneas - Formulario base
```

Total: ~3,550 líneas de código Object Pascal

15.2. Configuraciones del Sistema

Listing 14: Configuración del proyecto (.lpi)

```
<?xml version="1.0" encoding="UTF-8"?>
<CONFIG>
  <ProjectOptions>
    <Version Value="12"/>
    <General>
      <Title Value="EDDMail"/>
      <Scaled Value="True"/>
```



```

</General>
<BuildModes Count="2">
  <Item1 Name="Debug" Default="True"/>
  <Item2 Name="Release"/>
</BuildModes>
<PublishOptions>
  <Version Value="2"/>
  <UseFileFilters Value="True"/>
</PublishOptions>
<RunParams>
  <FormatVersion Value="2"/>
  <Modes Count="1">
    <Mode0 Name="default"/>
  </Modes>
</RunParams>
<RequiredPackages Count="1">
  <Item1>
    <PackageName Value="LCL"/>
  </Item1>
</RequiredPackages>
<Units Count="4">
  <Unit0>
    <Filename Value="EDDMail.lpr"/>
    <IsPartOfProject Value="True"/>
  </Unit0>
  <Unit1>
    <Filename Value="EstructurasDatos.pas"/>
    <IsPartOfProject Value="True"/>
  </Unit1>
  <Unit2>
    <Filename Value="CorreoManager.pas"/>
    <IsPartOfProject Value="True"/>
  </Unit2>
  <Unit3>
    <Filename Value="InterfazGTK.pas"/>
    <IsPartOfProject Value="True"/>
  </Unit3>
</Units>
</ProjectOptions>
</CONFIG>

```

15.3. Scripts de Automatización

Listing 15: Script de compilación automática

```

#!/bin/bash
# build.sh - Script de compilación para EDDMail

```

```

echo "====- Compilaci n -EDDMail-===="
echo "Verificando dependencias..."

# Verificar Free Pascal
if ! command -v fpc &> /dev/null; then
    echo "Error: -Free-Pascal-no-est -instalado"
    exit 1
fi

# Verificar GTK
if ! pkg-config --exists gtk+-2.0; then
    echo "Error: -GTK2-no-est -instalado"
    exit 1
fi

# Verificar Graphviz
if ! command -v dot &> /dev/null; then
    echo "Advertencia: -Graphviz-no-est -instalado-(reportes-no-funcionar"
fi

echo "Compilando proyecto..."
cd src/

# Compilar con opciones de depuraci n
fpc -dDEBUG -Fu../lib -Fi../lib -FE../bin -o../bin/EDDMail EDDMail.lpr

if [ $? -eq 0 ]; then
    echo "      -Compilaci n-exitosa"
    echo "Ejecutable-generado-en:-bin/EDDMail"

    # Crear directorios necesarios
    mkdir -p ../reportes/Root-Reportes
    mkdir -p ../datos

    echo "      -Directorios-creados"
else
    echo "      -Error-en-la-compilaci n"
    exit 1
fi

```

Listing 16: Script de limpieza

```

#!/bin/bash
# clean.sh - Limpieza de archivos temporales

echo "Limpiando archivos temporales..."

# Eliminar archivos de compilaci n
rm -f src/*.o

```

```

rm -f src/*.ppu
rm -f src/*.compiled
rm -f src/*.res

# Eliminar ejecutables
rm -f bin/EDDMail

# Limpiar reportes (opcional)
read -p " Limpiar reportes generados?-(y/N):-" choice
case "$choice" in
    y|Y )
        rm -rf reportes/*
        mkdir -p reportes/Root-Reportes
        echo " Root-Reportes limpiados"
        ;;
    * )
        echo " Reportes conservados"
        ;;
esac

echo " Limpieza completada"

```

15.4. Estructura JSON Completa de Correos

Listing 17: Formato JSON para carga masiva de correos

```

1 {
2   "correos": [
3     {
4       "usuario_id": 1,
5       "bandeja_entrada": [
6         {
7           "id": 101,
8           "remitente": "teacher@edd.com",
9           "estado": "no leido",
10          "programado": "no",
11          "asunto": "Reunion pendiente",
12          "fecha": "2025-07-26",
13          "mensaje": "No olvides nuestra reuni n de ma ana"
14        },
15        {
16          "id": 102,
17          "remitente": "aux-luis@edd.com",
18          "estado": "leido",
19          "programado": "no",
20          "asunto": "Invitacion",
21          "fecha": "2025-07-28",
22          "mensaje": "Te invito a la reuni n de ma ana"

```

```

23     }
24   ]
25 },
26 {
27   "usuario_id": 2,
28   "bandeja_entrada": [
29     {
30       "id": 201,
31       "remitente": "noreply@edd.com",
32       "estado": "no leído",
33       "programado": "si",
34       "asunto": "Bienvenido al sistema de correo EDD",
35       "fecha": "2025-07-30",
36       "mensaje": "Bienvenido al sistema de correo EDD"
37     }
38   ]
39 }
40 ]
41 }

```

15.5. Casos de Prueba

ID	Caso de Prueba	Entrada	Resultado Esperado
TC001	Login usuario root	email: root@edd.com, pass: root123	Acceso a panel admin
TC002	Registro usuario nuevo	Datos válidos únicos	Usuario registrado
TC003	Registro email duplicado	Email existente	Error: Email ya existe
TC004	Enviar correo a contacto	Destinatario en contactos	Correo enviado
TC005	Enviar sin contacto	Destinatario no contacto	Error: Agregar contacto
TC006	Agregar contacto válido	Email usuario existente	Contacto agregado
TC007	Ordenar bandeja	Correos desordenados	Correos ordenados A-Z
TC008	Eliminar correo	Correo en bandeja	Correo movido a papelera
TC009	Programar correo	Fecha futura válida	Correo programado
TC010	Generar reporte	Usuario con datos	Archivos .dot y .png

Cuadro 5: Casos de prueba principales

15.6. Comandos de Graphviz

Listing 18: Comandos para generar reportes manualmente

```
# Generar PNG desde archivo DOT
dot -Tpng usuarios.dot -o usuarios.png

# Generar SVG (escalable)
dot -Tsvg relaciones.dot -o relaciones.svg

# Generar PDF
dot -Tpdf comunidades.dot -o comunidades.pdf

# Generar con layout específico
neato -Tpng matriz.dot -o matriz.png      # Para grafos no direccionados
circo -Tpng contactos.dot -o contactos.png # Layout circular
fdp -Tpng usuarios.dot -o usuarios.png     # Layout por fuerzas

# Verificar sintaxis DOT
dot -v usuarios.dot
```

16. Glosario Técnico

- **AFD:** Autómata Finito Determinista
- **DOT:** Lenguaje de descripción de grafos de Graphviz
- **FIFO:** First In, First Out (Primero en entrar, primero en salir)
- **FPC:** Free Pascal Compiler
- **GTK:** GIMP Toolkit (librería para interfaces gráficas)
- **JSON:** JavaScript Object Notation
- **LIFO:** Last In, First Out (Último en entrar, primero en salir)
- **MVC:** Model-View-Controller
- **Object Pascal:** Lenguaje de programación orientado a objetos basado en Pascal
- **Puntero:** Variable que almacena direcciones de memoria
- **Record:** Estructura de datos que agrupa campos relacionados
- **Unit:** Módulo de código en Pascal

17. Referencias

1. Free Pascal Documentation: <https://www.freepascal.org/docs.html>
2. Lazarus IDE Manual: <https://www.lazarus-ide.org/>
3. GTK2 Reference: <https://docs.gtk.org/gtk2/>
4. Graphviz Documentation: <https://graphviz.org/documentation/>
5. Object Pascal Language Guide
6. Data Structures and Algorithms in Pascal
7. JSON Processing in Free Pascal
8. Linux System Programming Guide

18. Información de Contacto y Soporte

18.1. Repositorio del Proyecto

- **GitHub:** https://github.com/JoseArt777/-EDD-1S2025_02100305.git **Branch principal :** *main*
- **Releases:** Tags con versiones estables
- **Issues:** Para reportar bugs y solicitar features

18.2. Documentación Adicional

- **Manual de Usuario:** Guía para usuarios finales
- **Manual de Integración:** Documentación para trabajo en grupo
- **README.md:** Información básica del proyecto
- **CHANGELOG.md:** Historial de cambios entre versiones

18.3. Créditos

- **Desarrollador:** José Alexander López López
- **Curso:** Estructuras de Datos
- **Institución:** Universidad de San Carlos de Guatemala
- **Fecha:** 3 de septiembre de 2025
- **Versión del documento:** 1.0