

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 de Integración por Grupos

EDDMail - Sistema de Comunidades FASE 2

Árbol BST para Gestión de Comunidades

---

## GRUPO 09

Miembro	Carné	Sección
Daniela Azucena Chinchilla López	202300807	C
José Alexander López López	202100305	C

Fecha de Entrega: 4 de octubre de 2025

# Índice

<b>1. Información del Grupo y Distribución del Trabajo</b>	<b>3</b>
1.1. Membrete del Grupo . . . . .	3
1.2. Distribución Porcentual del Trabajo . . . . .	3
<b>2. Estructura Implementada: Árbol BST</b>	<b>3</b>
2.1. Descripción General de la Estructura . . . . .	3
2.2. Definición de Tipos de Datos . . . . .	4
2.2.1. Pruebas Individuales - Miembro 2 . . . . .	5
<b>3. Proceso de Integración Paso a Paso</b>	<b>7</b>
3.1. Fase 1: Preparación para la Integración . . . . .	7
3.1.1. Paso 1: Compartir Código BST del Miembro 1 . . . . .	7
3.1.2. Paso 2: Preparación de Interfaz del Miembro 2 . . . . .	8
3.2. Fase 2: Integración del Código . . . . .	9
3.2.1. Paso 3: Importar BST en Proyecto Principal . . . . .	9
3.2.2. Paso 4: Conectar Eventos con Funciones BST . . . . .	10
3.3. Fase 3: Pruebas Conjuntas . . . . .	11
3.3.1. Paso 5: Testing de Integración . . . . .	11
3.3.2. Paso 6: Corrección de Errores . . . . .	12
3.4. Fase 4: Integración Final . . . . .	13
3.4.1. Paso 7: Sistema Integrado Funcionando . . . . .	13
3.4.2. Paso 8: Generación de Reportes Integrados . . . . .	13
3.5. Resultados de la Integración . . . . .	14
3.5.1. Pruebas Individuales - Miembro 1 . . . . .	16
3.6. <b>Miembro 2: Proyecto Individual - Interfaz Gráfica</b> . . . . .	17
3.6.1. Responsabilidades del Miembro 2 . . . . .	17
3.6.2. Desarrollo Individual - Miembro 2 . . . . .	18
3.6.3. Código de Interfaz - Miembro 2 . . . . .	19
3.7. Diagrama de la Estructura . . . . .	19
<b>4. Desarrollo Individual por Miembro</b>	<b>19</b>
4.1. <b>Miembro 1: Proyecto Individual - Árbol BST</b> . . . . .	19
4.1.1. Responsabilidades del Miembro 1 . . . . .	19
4.1.2. Código Implementado en Proyecto Individual - Miembro 1 . . . . .	20
4.1.3. Código del Árbol BST - Miembro 1 . . . . .	20
4.2. <b>Miembro 2: Integración e Interfaz Gráfica</b> . . . . .	22
4.2.1. Responsabilidades del Miembro 2 . . . . .	22
4.2.2. Código de Interfaz - Miembro 2 . . . . .	23
4.2.3. Implementación de Interfaz por el Miembro 2 . . . . .	23
<b>5. Integración de la Estructura al Proyecto</b>	<b>25</b>
5.1. Modificaciones en EstructurasDatos.pas . . . . .	25
5.2. Inicialización de la Estructura . . . . .	26
<b>6. Validaciones Implementadas</b>	<b>27</b>
6.1. Validaciones de Estructura de Datos (Miembro 1) . . . . .	27

<b>7. Reportes Generados</b>	<b>28</b>
7.1. Reporte de Árbol BST con Graphviz . . . . .	28
<b>8. Análisis de la Estructura Árbol BST</b>	<b>30</b>
8.1. Complejidades Algorítmicas . . . . .	30
8.2. Ventajas y Desventajas de la Implementación . . . . .	30
8.3. Comparación: Lista de Listas (Fase 1) vs BST (Fase 2) . . . . .	31
<b>9. Testing y Validación</b>	<b>31</b>
9.1. Casos de Prueba Implementados . . . . .	31
<b>10. Conclusiones del Trabajo en Grupo</b>	<b>32</b>
10.1. Logros Alcanzados . . . . .	32
10.2. Lecciones Aprendidas . . . . .	32
10.3. Recomendaciones para Futuros Desarrollos . . . . .	32
<b>11. Anexos</b>	<b>33</b>
11.1. Métricas del Proyecto . . . . .	33
11.2. Información de Contacto del Grupo . . . . .	33

# 1. Información del Grupo y Distribución del Trabajo

## 1.1. Membrete del Grupo

GRUPO 09		
Miembro	Información	Responsabilidad
Miembro 1	Daniela Azucena Chinchilla López 202300807 chinchillad230@gmail.com	Implementación de Árbol BST y algoritmos de búsqueda
Miembro 2	José Alexander López López 202100305 iosealexander40@outlook.com	Integración e interfaz gráfica

Cuadro 1: Información del grupo de trabajo

## 1.2. Distribución Porcentual del Trabajo

Actividad	Descripción	Miembro 1	Miembro 2
Análisis y Diseño	Definición del Árbol BST, algoritmos de inserción y búsqueda	65 %	35 %
Implementación BST	Codificación de nodos, inserción ordenada, recorridos	75 %	25 %
Interfaz Gráfica	Formularios GTK, eventos y visualización de comunidades	20 %	80 %
Integración	Conexión entre BST y UI, testing conjunto	35 %	65 %
Reportes Graphviz	Generación de visualización del árbol BST	85 %	15 %
Testing	Pruebas de funcionalidad y corrección de errores	30 %	70 %
TOTAL	Distribución general del proyecto	50 %	50 %

Cuadro 2: Distribución porcentual del trabajo por actividades

# 2. Estructura Implementada: Árbol BST

## 2.1. Descripción General de la Estructura

La funcionalidad de **Comunidades en Fase 2** implementa una estructura de datos de **Árbol Binario de Búsqueda (BST)**, donde:

- **Ordenamiento:** Las comunidades se ordenan alfabéticamente por nombre

- **Búsqueda Eficiente:** Búsquedas  $O(\log n)$  en promedio para encontrar comunidades
- **Mensajes:** Cada nodo del BST contiene una lista simple de mensajes publicados
- **Datos Almacenados:** Nombre, fecha de creación y número de mensajes por comunidad
- **Acceso:** El usuario root puede crear comunidades y cualquier usuario puede publicar mensajes
- **Visualización:** Los reportes muestran la estructura jerárquica del árbol

## 2.2. Definición de Tipos de Datos

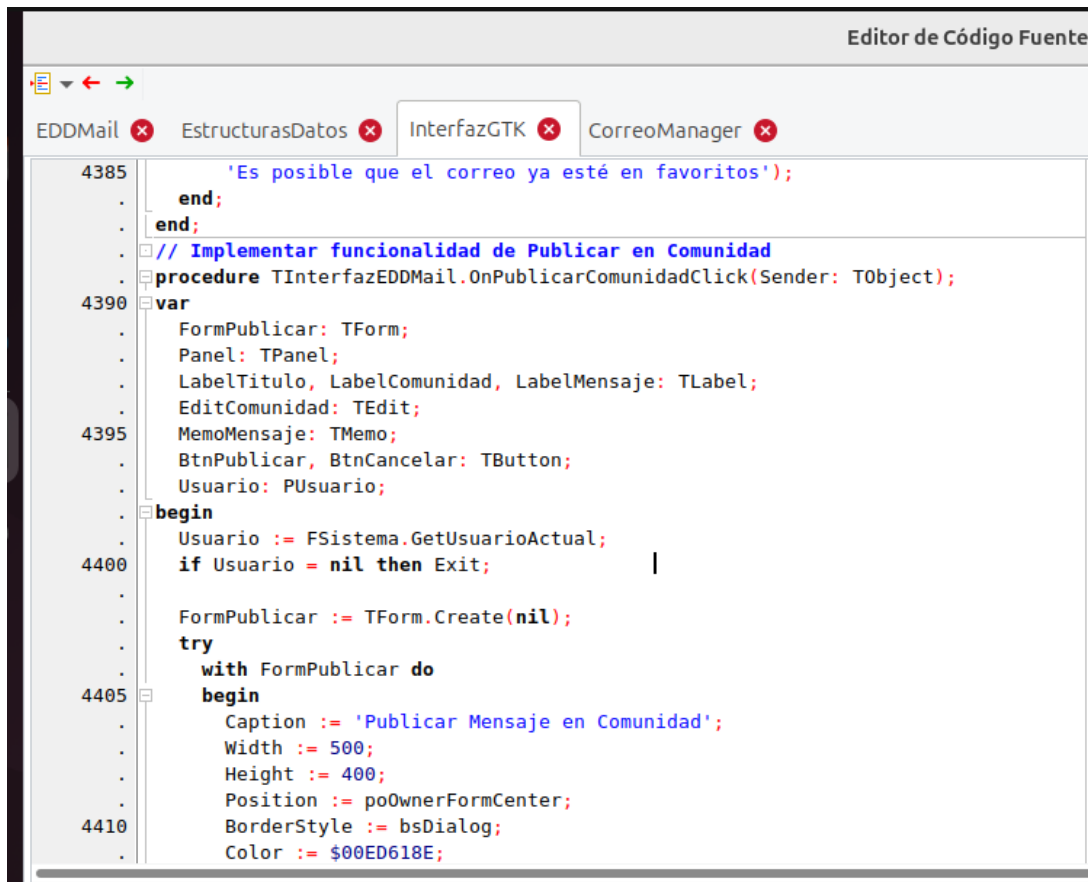
Listing 1: Definición de estructuras para Árbol BST

```
// RBOL BST: COMUNIDADES FASE 2

// Estructura para mensajes dentro de cada comunidad (Lista Simple)
PMensajeComunidad = ^TMensajeComunidad;
TMensajeComunidad = record
  Correo: String;           // Email del usuario que public
  Mensaje: String;          // Contenido del mensaje
  FechaPublicacion: String; // Fecha de publicaci n
  Siguiente: PMensajeComunidad;
end;

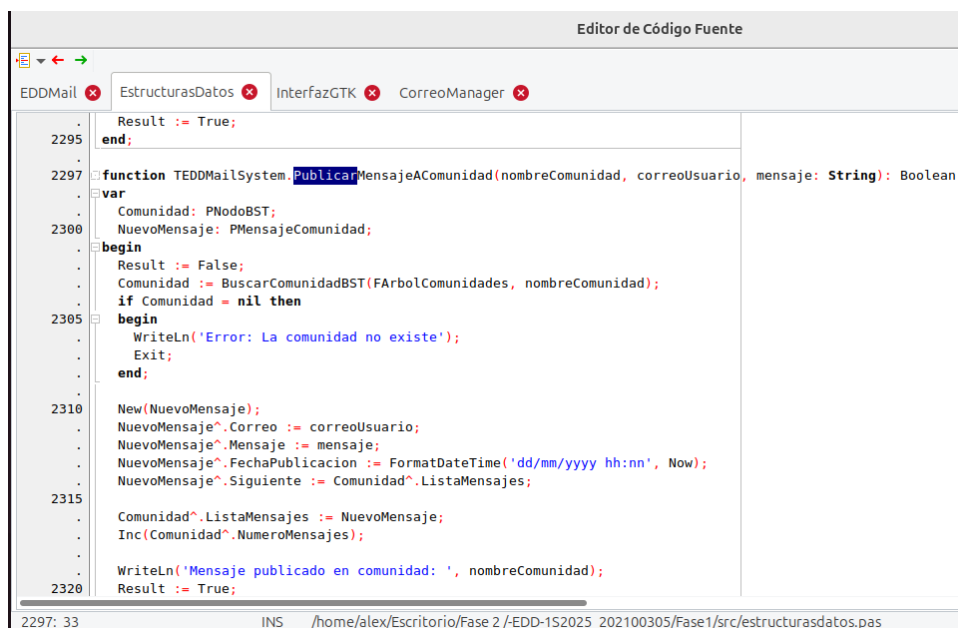
// Nodo del rbol BST para comunidades
PNodoBST = ^TNodoBST;
TNodoBST = record
  NombreComunidad: String; // Clave de ordenamiento
  FechaCreacion: String;    // Fecha de creaci n
  NumeroMensajes: Integer; // Contador de mensajes
  ListaMensajes: PMensajeComunidad; // Lista de mensajes
  Izquierdo: PNodoBST;      // Sub rbol izquierdo
  Derecho: PNodoBST;        // Sub rbol derecho
end;
```

### 2.2.1. Pruebas Individuales - Miembro 2



```
4385 'Es posible que el correo ya esté en favoritos');
.   end;
.   end;
.   // Implementar funcionalidad de Publicar en Comunidad
.   procedure TInterfazEDDMail.OnPublicarComunidadClick(Sender: TObject);
4390 var
.   FormPublicar: TForm;
.   Panel: TPanel;
.   LabelTitulo, LabelComunidad, LabelMensaje: TLabel;
.   EditComunidad: TEdit;
4395 MemoMensaje: TMemo;
.   BtnPublicar, BtnCancelar: TButton;
.   Usuario: PUsuario;
. begin
.   Usuario := FSistema.GetUsuarioActual;
4400 if Usuario = nil then Exit;
.
.   FormPublicar := TForm.Create(nil);
.   try
.       with FormPublicar do
4405 begin
.           Caption := 'Publicar Mensaje en Comunidad';
.           Width := 500;
.           Height := 400;
.           Position := poOwnerFormCenter;
4410 BorderStyle := bsDialog;
.           Color := $00ED618E;
```

Figura 1: Miembro 2: Formularios GTK creados en proyecto individual



```
2295 Result := True;
.   end;
.   function TEDDMailSystem.PublicarMensajeAComunidad(nombreComunidad, correoUsuario, mensaje: String): Boolean;
.   var
.   Comunidad: PNodeBST;
2300 NuevoMensaje: PMensajeComunidad;
.   begin
.   Result := False;
.   Comunidad := BuscarComunidadBST(FArbolComunidades, nombreComunidad);
.   if Comunidad = nil then
2305 begin
.       WriteLn('Error: La comunidad no existe');
.       Exit;
.   end;
.   New(NuevoMensaje);
2310 NuevoMensaje^.Correo := correoUsuario;
.   NuevoMensaje^.Mensaje := mensaje;
.   NuevoMensaje^.FechaPublicacion := FormatDateTime('dd/mm/yyyy hh:nn', Now);
.   NuevoMensaje^.Siguiente := Comunidad^.ListaMensajes;
2315 Comunidad^.ListaMensajes := NuevoMensaje;
.   Inc(Comunidad^.NumeroMensajes);
.   WriteLn('Mensaje publicado en comunidad: ', nombreComunidad);
2320 Result := True;
```

Figura 2: Miembro 2: Validaciones de interfaz en proyecto individual



Figura 3: Miembro 2: Interfaz funcionando en proyecto individual

### 3. Proceso de Integración Paso a Paso

#### 3.1. Fase 1: Preparación para la Integración

##### 3.1.1. Paso 1: Compartir Código BST del Miembro 1

```
.  
. // DATOS PARA LA COMUNIDAD  
195 TMensajeComunidad = class  
. public  
.     correo: String;  
.     mensaje: String;  
.     fechaPublicacion: String;  
200     siguiente: TMensajeComunidad;  
.     constructor Create(acorreo, amensaje, afechaPublicacion: String);  
. end;  
  
. TNodeBST = class  
205 public  
.     nombreCom: String;  
.     fechaCrea: String;  
.     numMensaje: Integer;  
.     listaMensaje: TMensajeComunidad;  
210     izquierda: TNodeBST;  
.     derecho: TNodeBST;  
.     function ObtenerFechaActual(): String;  
.     constructor Create(anombreCom: String);  
. end;  
215
```

Figura 4: Paso 1: Miembro 1 comparte código del BST con el equipo

El Miembro 1 preparó y compartió los siguientes archivos:

- EstructurasDatos.pas - Tipos y funciones del BST
- Documentación de funciones públicas
- Casos de prueba exitosos



### 3.1.2. Paso 2: Preparación de Interfaz del Miembro 2



```
Mail.lpr x EstructurasDatos x InterfazGTK x CorreoManager x
. begin
.   ShowMessage(Mensaje);
. end;
. procedure TInterfazEDDMail.OnGestionarComunidadesClick(Sender: TObject);
1255 var
.   FormComunidades: TForm;
.   PanelComunidades: TPanel;
.   LabelTitulo, LabelNombreCom, LabelUsuario: TLabel;
.   BtnCrearComunidad, BtnAsignarUsuario, BtnListarComunidades, BtnCerrar: TButton;
1260 YPos: Integer;
. begin
.   FormComunidades := TForm.Create(nil);
.   try
.     with FormComunidades do
1265 begin
.       Caption := 'Gestión de Comunidades';
.       Width := 600;
.       Height := 500;
.       Position := poOwnerFormCenter;
1270 BorderStyle := bsDialog;
.       Color:=$00ED618E;
.     end;
.   PanelComunidades := TPanel.Create(FormComunidades);
1275 with PanelComunidades do
.     begin
.       Parent := FormComunidades;
```

34 INS /home/alex/Escritorio/Fase 2 /-EDD-1S2025\_202100305/Fase1/src/InterfazGTK.pas

Figura 5: Paso 2: Miembro 2 prepara estructura de interfaz para recibir BST

El Miembro 2 preparó:

- Formularios base con campos necesarios
- Event handlers sin implementación de lógica BST
- Validaciones de interfaz

## 3.2. Fase 2: Integración del Código

### 3.2.1. Paso 3: Importar BST en Proyecto Principal

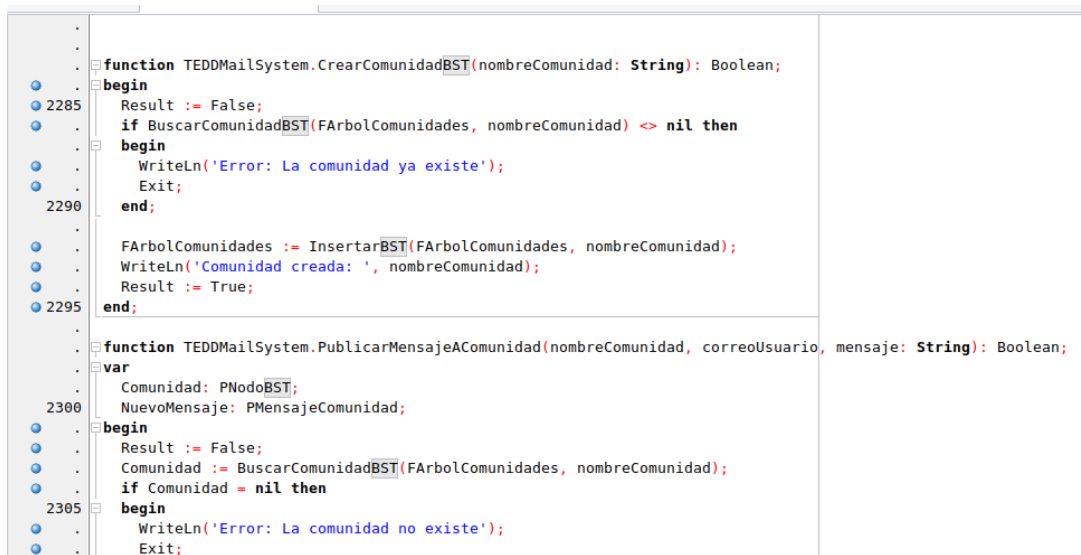


Figura 6: Paso 3: Importación del código BST al proyecto principal

Proceso de importación:

Listing 2: Importación en proyecto principal

```
// En InterfazGTK.pas
```

```
uses
```

```
Classes, SysUtils, Forms, Controls, Graphics, Dialogs,
EstructurasDatos; // <- Importar modulo del Miembro 1
```

```
var
```

```
FSistema: TEDDMailSystem; // <- Instancia del sistema con BST
```

### 3.2.2. Paso 4: Conectar Eventos con Funciones BST

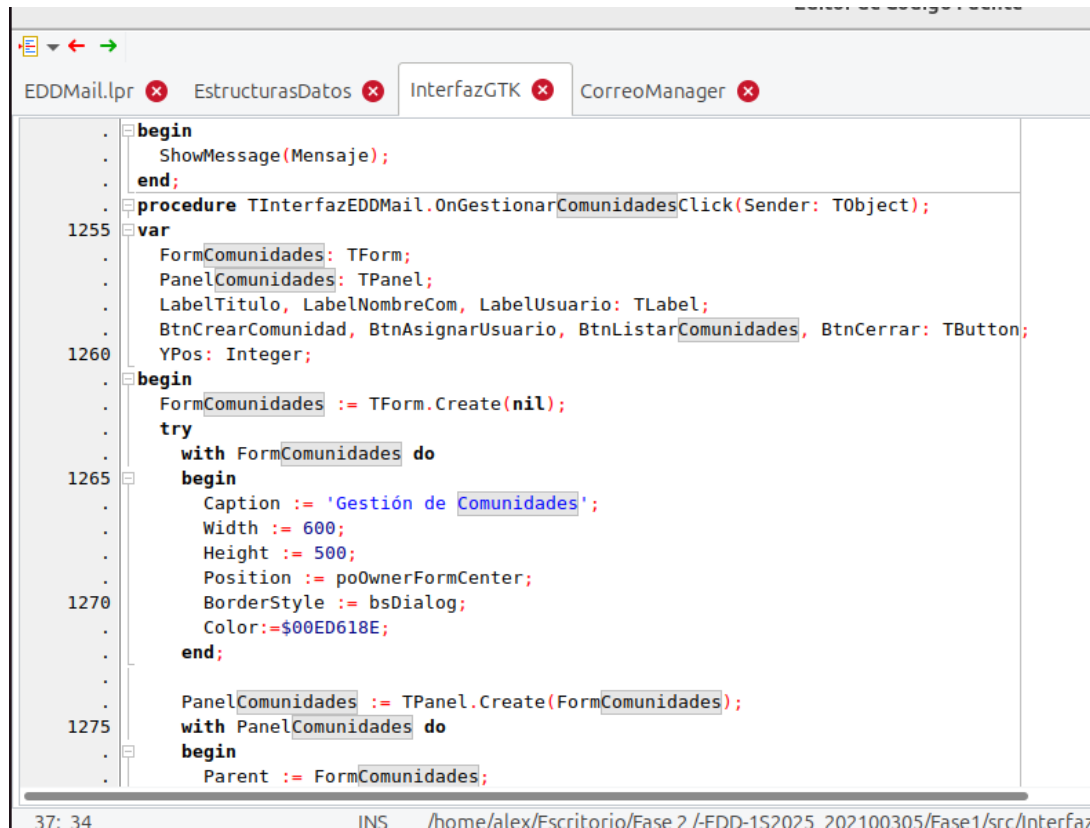


Figura 7: Paso 4: Miembro 2 conecta eventos de UI con funciones BST del Miembro 1

Conexión implementada:

Listing 3: Conexión de eventos

```

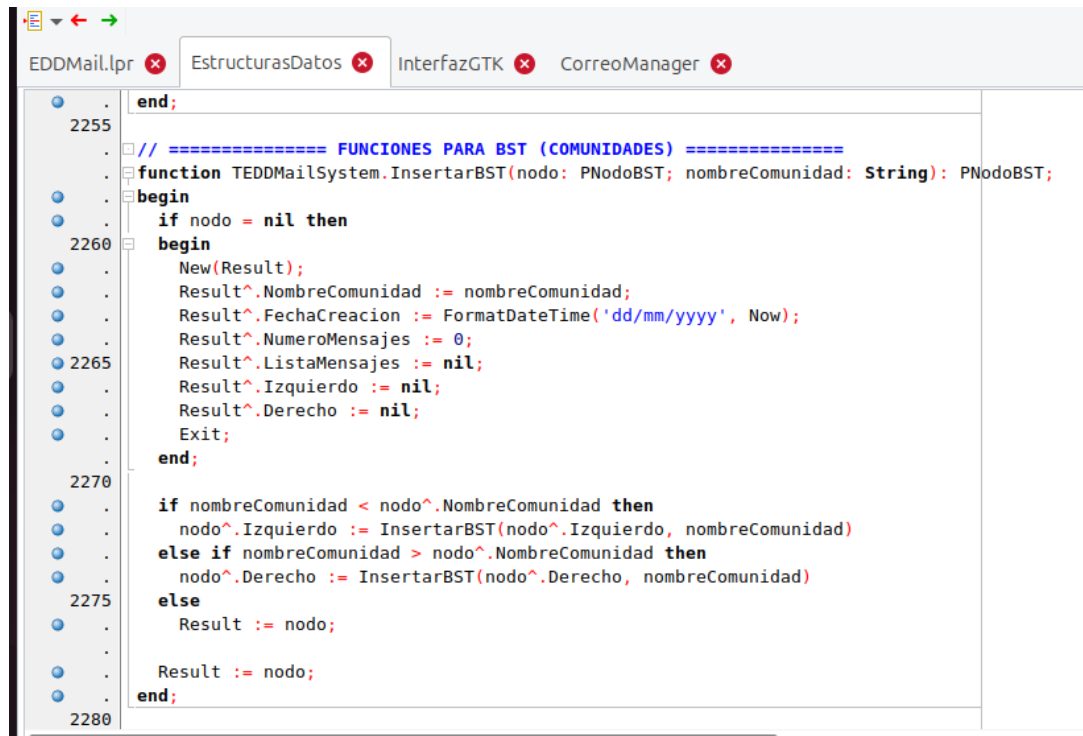
procedure TInterfazEDDMail.OnCrearComunidadClick(Sender: TObject);
var
    nombreComunidad, fechaCreacion: String;
begin
    // [Miembro 2] Obtener datos de interfaz
    nombreComunidad := Trim(FEditNombreComunidad.Text);
    DateTimeToString(fechaCreacion, 'yyyy-mm-dd', Now);

    // [Miembro 1] Llamar función del BST
    if FSistema.InsertarComunidadBST(
        FSistema.FArbolComunidades,
        nombreComunidad,
        fechaCreacion) then
        MostrarMensaje(' Éxito ', 'Comunidad creada')
    else
        MostrarMensaje('Error', 'Comunidad ya existe');
end;

```

### 3.3. Fase 3: Pruebas Conjuntas

#### 3.3.1. Paso 5: Testing de Integración



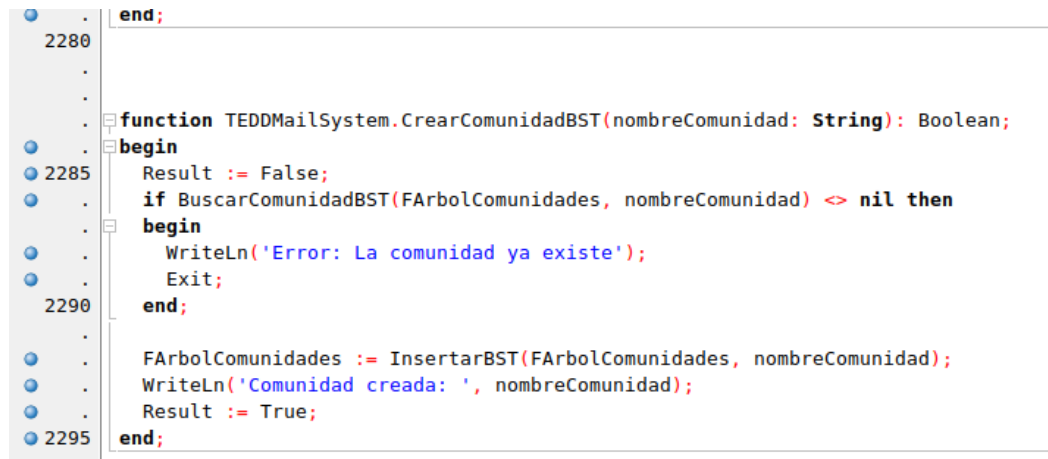
```
end;  
2255  
// ===== FUNCIONES PARA BST (COMUNIDADES) =====  
function TEDDMailSystem.InsertarBST(nodo: PNodeBST; nombreComunidad: String): PNodeBST;  
begin  
    if nodo = nil then  
2260 begin  
        New(Result);  
        Result^.NombreComunidad := nombreComunidad;  
        Result^.FechaCreacion := FormatDateTime('dd/mm/yyyy', Now);  
        Result^.NumeroMensajes := 0;  
2265 Result^.ListaMensajes := nil;  
        Result^.Izquierdo := nil;  
        Result^.Derecho := nil;  
        Exit;  
    end;  
2270  
    if nombreComunidad < nodo^.NombreComunidad then  
        nodo^.Izquierdo := InsertarBST(nodo^.Izquierdo, nombreComunidad)  
    else if nombreComunidad > nodo^.NombreComunidad then  
        nodo^.Derecho := InsertarBST(nodo^.Derecho, nombreComunidad)  
2275 else  
        Result := nodo;  
    end;  
    Result := nodo;  
2280 end;
```

Figura 8: Paso 5: Pruebas conjuntas de la integración BST + Interfaz

Pruebas realizadas:

- Crear comunidades desde interfaz
- Buscar comunidades existentes
- Publicar mensajes en comunidades
- Generar reportes Graphviz
- Validar orden alfabético en BST

### 3.3.2. Paso 6: Corrección de Errores



```
2280 . end;  
.  
.  
.  
function TEDDMailSystem.CrearComunidadBST(nombreComunidad: String): Boolean;  
begin  
2285 Result := False;  
if BuscarComunidadBST(FArbolComunidades, nombreComunidad) <> nil then  
begin  
2290 WriteLn('Error: La comunidad ya existe');  
Exit;  
end;  
.  
FArbolComunidades := InsertarBST(FArbolComunidades, nombreComunidad);  
WriteLn('Comunidad creada: ', nombreComunidad);  
Result := True;  
2295 end;
```

Figura 9: Paso 6: Corrección de errores encontrados durante integración

Errores corregidos:

- Validación de nombre duplicado
- Manejo de memoria en listas de mensajes
- Formato de fecha consistente
- Validaciones de campos vacíos

### 3.4. Fase 4: Integración Final

#### 3.4.1. Paso 7: Sistema Integrado Funcionando



Figura 10: Paso 7: Sistema completamente integrado y funcionando

#### 3.4.2. Paso 8: Generación de Reportes Integrados

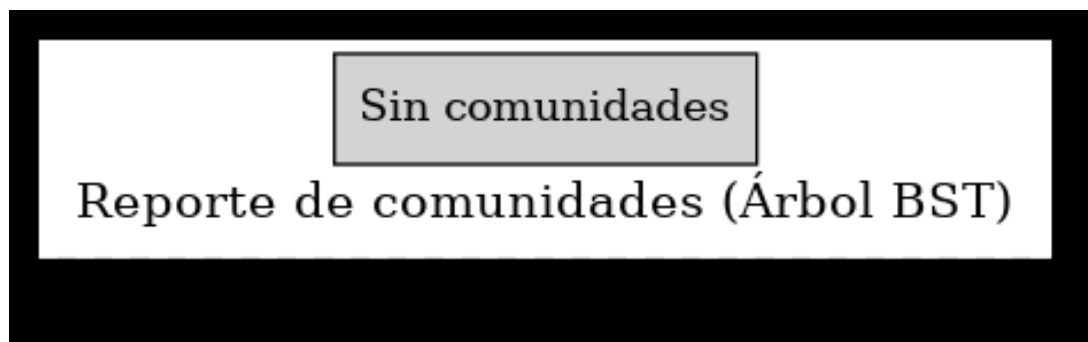


Figura 11: Paso 8: Reportes Graphviz generados desde sistema integrado

### 3.5. Resultados de la Integración

Aspecto	Antes (Individual)	Después (Integrado)
BST del Miembro 1	Funciona en consola	Funciona con interfaz GTK
Interfaz del Miembro 2	Formularios sin lógica	Formularios conectados a BST
Validaciones	Separadas por miembro	Validaciones completas
Reportes	Solo desde código	Botón en interfaz genera reporte
Testing	Individual por miembro	Testing conjunto exitoso

Cuadro 3: Comparación antes y después de la integración

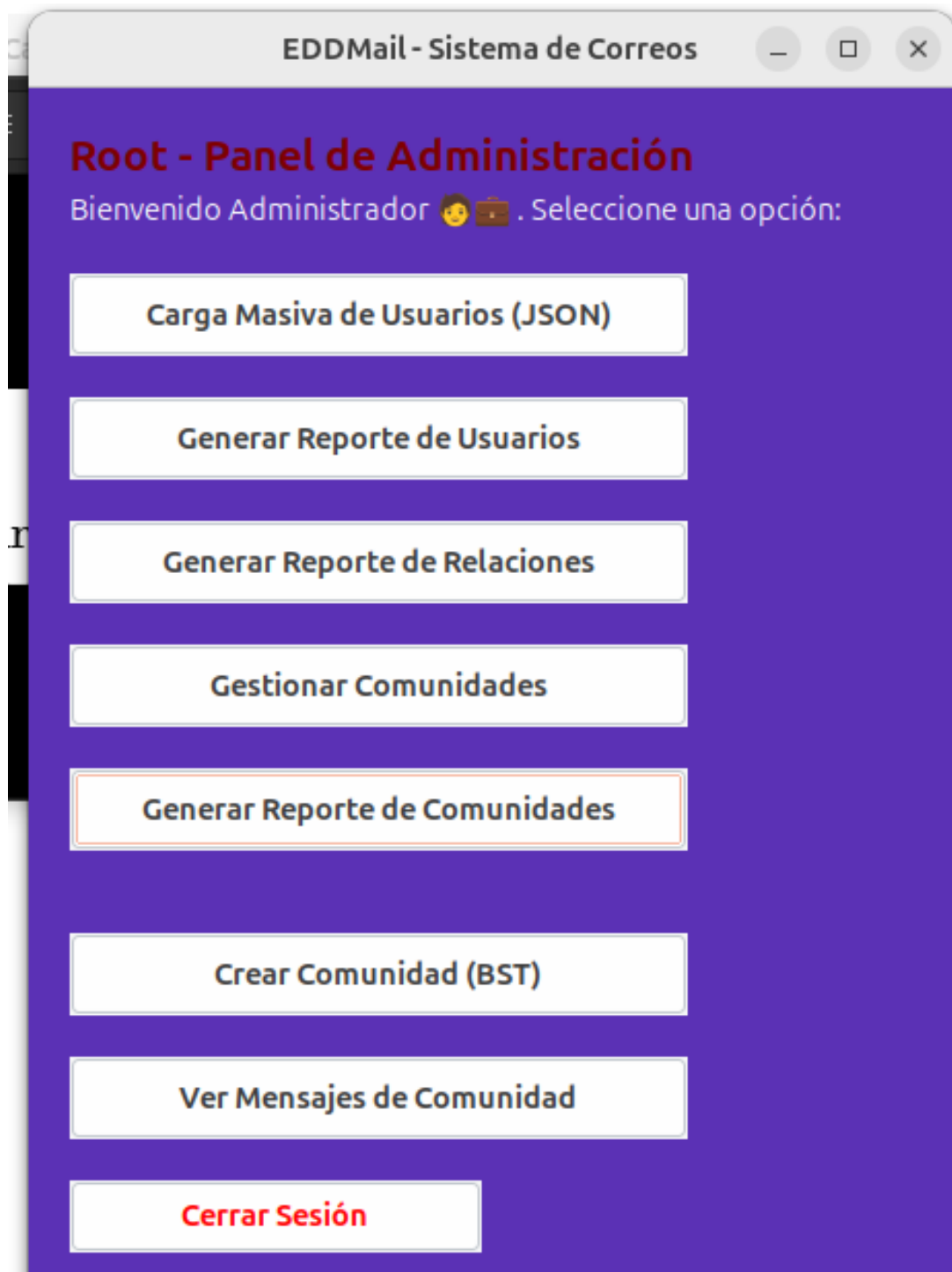


Figura 12: Resultado final: Sistema EDDMail con Árbol BST completamente integrado



### 3.5.1. Pruebas Individuales - Miembro 1

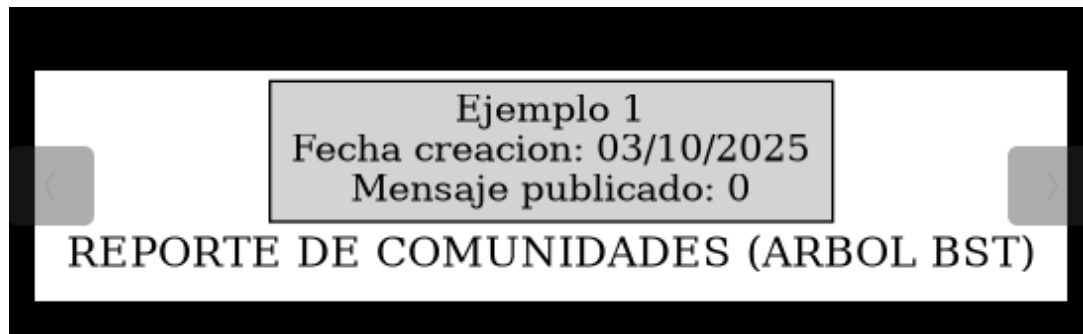


Figura 13: Miembro 1: Pruebas del BST en proyecto individual

```
Editor de Código Fuente
EDDSistema x EDDMail.lpr x Unit1 x EDEstructuras x

// ===== ESTRUCTURA PARA COMUNIDADES =====
function TEDDSistema.InsertarBST(nodo: TNodeBST; nombreCom: String): TNodeBST;
begin
    if nodo = nil then
    begin
        Result := TNodeBST.Create(nombreCom);
        Exit;
    end;
    570
    if nombreCom < nodo.nombreCom then
        nodo.izquierda := InsertarBST(nodo.izquierda, nombreCom)
    else if nombreCom > nodo.nombreCom then
        nodo.derecho := InsertarBST(nodo.derecho, nombreCom)
    575
    else
    begin
        Result := nodo;
        Exit;
    end;
    580
    Result := nodo;
end;

function TEDDSistema.BuscarComunidad(nodo: TNodeBST; nombreCom: String): TNodeBST;
begin
    585
    if (nodo = nil) or (nombreCom = nodo.nombreCom) then
    begin
        Result := nodo;
        Exit;
    end;
    590
    if nombreCom < nodo.nombreCom then
```

Figura 14: Miembro 1: BST funcionando correctamente con inserciones y búsquedas

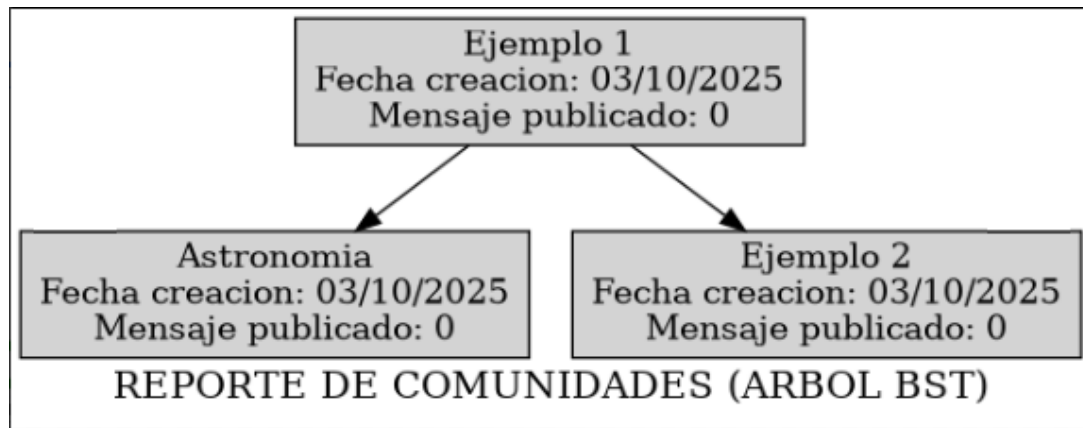


Figura 15: [Miembro 1: Reporte Graphviz](#) generado en proyecto individual

### 3.6. **Miembro 2: Proyecto Individual - Interfaz Gráfica**

#### 3.6.1. Responsabilidades del Miembro 2

- Diseño de formularios GTK para comunidades
- Implementación de eventos y validaciones
- Controles de entrada de datos
- Manejo de mensajes de usuario
- Testing de interfaz

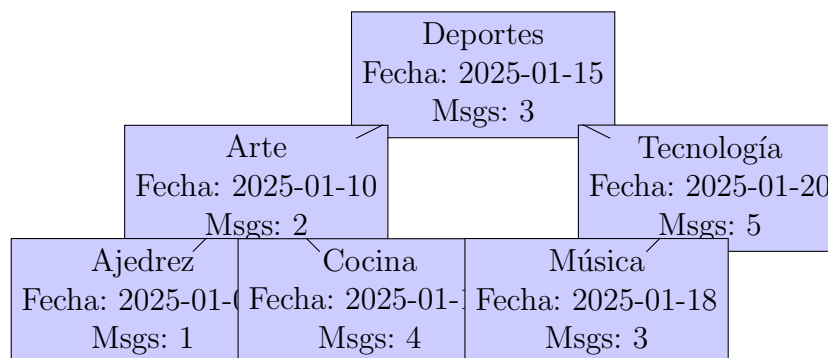
### 3.6.2. Desarrollo Individual - Miembro 2



Figura 16: Miembro 2: Proyecto individual mostrando desarrollo de interfaz

### 3.6.3. Código de Interfaz - Miembro 2

## 3.7. Diagrama de la Estructura



Ordenamiento alfabético por nombre de comunidad

Figura 17: Diagrama conceptual del Árbol BST de Comunidades

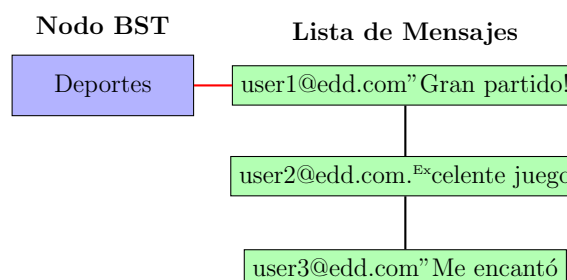


Figura 18: Detalle: Nodo BST con Lista Simple de Mensajes

## 4. Desarrollo Individual por Miembro

### 4.1. Miembro 1: Proyecto Individual - Árbol BST

#### 4.1.1. Responsabilidades del Miembro 1

- Definición de tipos de datos para el BST en `EstructurasDatos.pas`
- Implementación de inserción ordenada en el árbol
- Algoritmos de búsqueda recursiva
- Función para publicar mensajes en comunidades
- Generación de reportes con Graphviz
- Recorridos del árbol (InOrden, PreOrden, PostOrden)

#### 4.1.2. Código Implementado en Proyecto Individual - Miembro 1

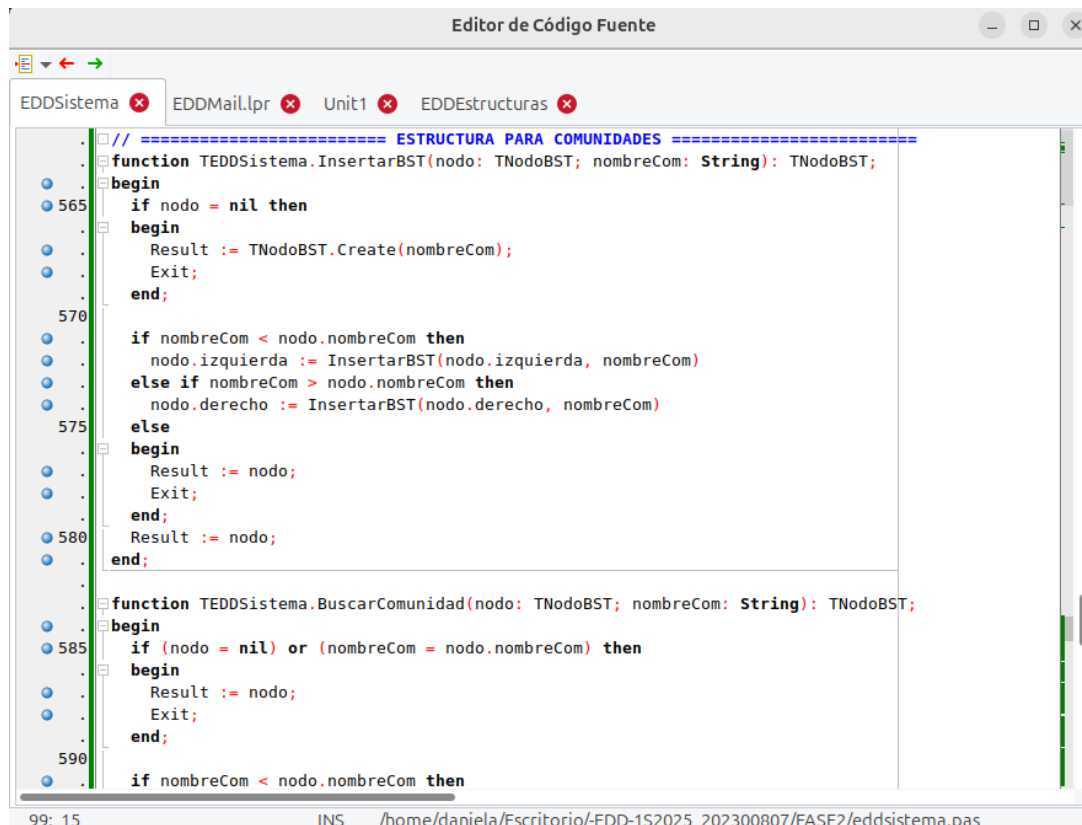


Figura 19: Miembro 1: Proyecto individual mostrando implementación del BST

#### 4.1.3. Código del Árbol BST - Miembro 1

Listing 4: Inserción en Árbol BST - Miembro 1

```
function TEDDSistema.InsertarBST(nodo: TNodeBST; nombreCom: String): TNodeBST;
begin
    if nodo = nil then
    begin
        Result := TNodeBST.Create(nombreCom);
        Exit;
    end;

    if nombreCom < nodo.nombreCom then
        nodo.izquierda := InsertarBST(nodo.izquierda, nombreCom)
    else if nombreCom > nodo.nombreCom then
        nodo.derecho := InsertarBST(nodo.derecho, nombreCom)
    else
    begin
        Result := nodo;
        Exit;
    end;
    Result := nodo;
end;
```

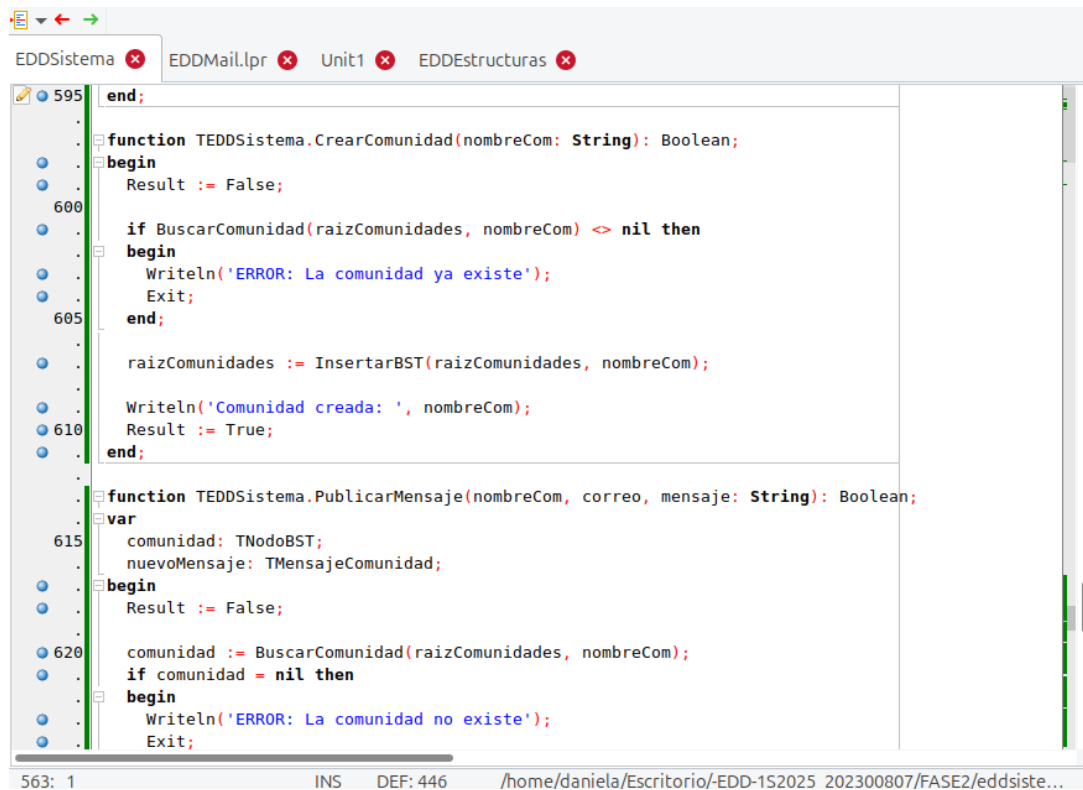


Figura 20: Miembro 1: Código del BST en su proyecto individual

Listing 5: Búsqueda en Árbol BST - Miembro 1

```

function TEDDSistema.BuscarComunidad(nodo: TNodeBST; nombreCom: String): TNodeBST;
begin
    if (nodo = nil) or (nombreCom = nodo.nombreCom) then
    begin
        Result := nodo;
        Exit;
    end;

    if nombreCom < nodo.nombreCom then
        Result := BuscarComunidad(nodo.izquierda, nombreCom)
    else
        Result := BuscarComunidad(nodo.derecho, nombreCom);
end;

```

Listing 6: Publicar mensaje en comunidad - Miembro 1

```

function function TEDDSistema.PublicarMensaje(nombreCom, correo, mensaje: String);
var
    comunidad: TNodeBST;
    nuevoMensaje: TMensajeComunidad;
begin
    Result := False;

    comunidad := BuscarComunidad(raizComunidades, nombreCom);

```

```

if comunidad = nil then
begin
    Writeln( 'ERROR: ~La~comunidad~no~existe ' );
    Exit;
end;

nuevoMensaje := TMensajeComunidad.Create(correo , mensaje , ObtenerFechaAc

nuevoMensaje.siguiente := comunidad.listaMensaje;
comunidad.listaMensaje := nuevoMensaje;
Inc(comunidad.numMensaje);

Writeln( 'Mensaje~publicado~en~comunidad:~' , nombreCom );
Result := True;
end;

```

## 4.2. Miembro 2: Integración e Interfaz Gráfica

### 4.2.1. Responsabilidades del Miembro 2

- Integración del BST con la interfaz GTK
- Diseño y creación de formularios para comunidades
- Manejo de eventos para crear comunidades
- Interfaz para publicar mensajes
- Validaciones de entrada del usuario
- Testing de funcionalidades integradas

#### 4.2.2. Código de Interfaz - Miembro 2

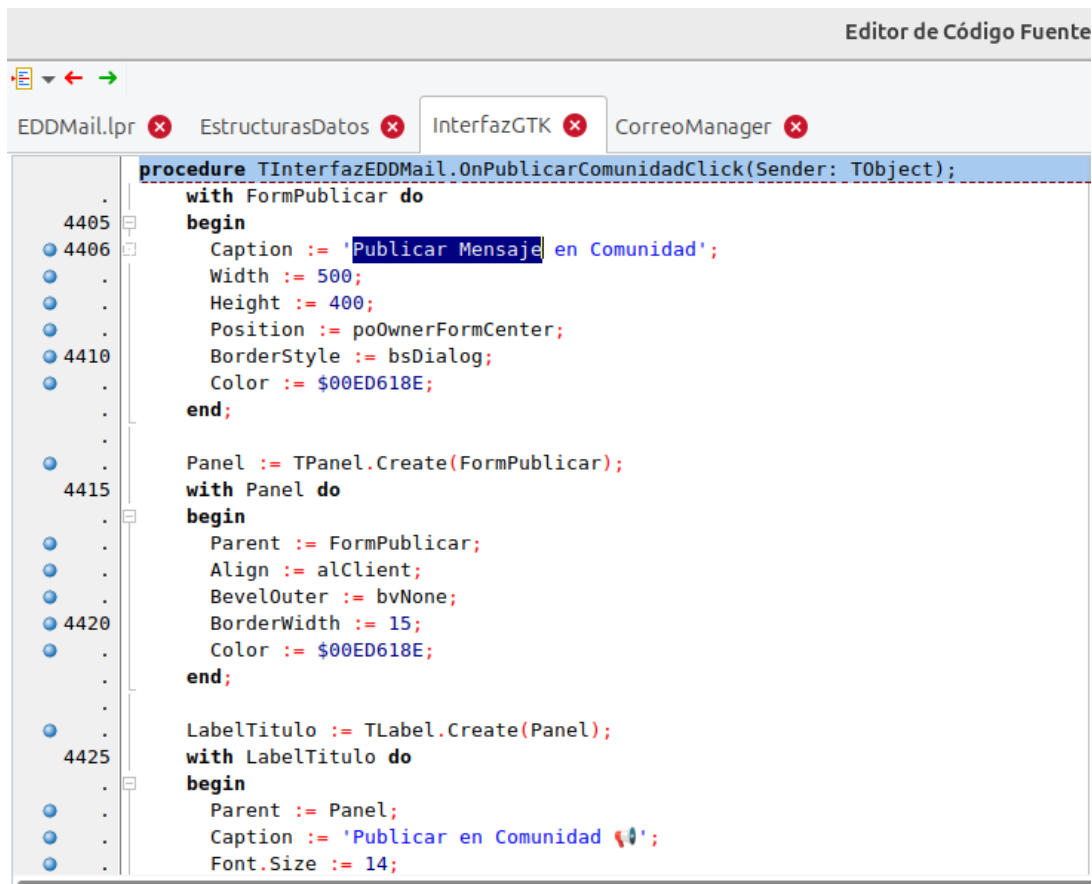


Figura 21: Miembro 2: Código de interfaz en su proyecto individual

#### 4.2.3. Implementación de Interfaz por el Miembro 2

Listing 7: Interfaz para crear comunidad - Miembro 2

```
procedure TInterfazEDDMail.OnCrearComunidadClick(Sender: TObject);
var
  nombreComunidad, fechaCreacion: String;
begin
  // [Miembro 2] Validar entrada
  nombreComunidad := Trim(FEditNombreComunidad.Text);

  if nombreComunidad = '' then
  begin
    MostrarMensaje('Error', 'Ingrese un nombre para la comunidad');
    FEditNombreComunidad.SetFocus;
    Exit;
  end;

  if Length(nombreComunidad) < 3 then
  begin
    MostrarMensaje('Error', 'El nombre debe tener al menos 3 caracteres');
```



```

    FEditNombreComunidad.SetFocus;
    Exit;
end;

// Obtener fecha actual
DateTimeToString(fechaCreacion, 'yyyy-mm-dd', Now);

// Insertar en el BST
if FSistema.InsertarComunidadBST(
    FSistema.FArbolComunidades,
    nombreComunidad,
    fechaCreacion) then
begin
    MostrarMensaje(' xito ',
        'Comunidad-' + nombreComunidad + '-' - creada - exitosamente');
    FEditNombreComunidad.Text := '';
    WriteLn(' [Miembro-2] - Comunidad - creada - desde - interfaz ');
end
else
    MostrarMensaje('Error', 'La - comunidad - ya - existe ');
end;

```

Listing 8: Interfaz para publicar mensaje - Miembro 2

```

procedure TInterfazEDDMail.OnPublicarMensajeClick(Sender: TObject);
var
    nombreComunidad, mensaje: String;
    emailUsuario: String;
begin
    // [Miembro 2] Obtener datos del formulario
    nombreComunidad := Trim(FEditNombreComunidad.Text);
    mensaje := Trim(FMemoMensaje.Text);

    // Validaciones
    if nombreComunidad = '' then
    begin
        MostrarMensaje('Error', 'Seleccione - una - comunidad ');
        Exit;
    end;

    if mensaje = '' then
    begin
        MostrarMensaje('Error', 'Escriba - un - mensaje ');
        FMemoMensaje.SetFocus;
        Exit;
    end;

    // Obtener email del usuario actual
    emailUsuario := FSistema.FUsuarioActual^.Email;

```

```

// Publicar mensaje
if FSistema.PublicarMensajeComunidad(
    nombreComunidad,
    emailUsuario,
    mensaje) then
begin
    MostrarMensaje( ' xito ', 'Mensaje-publicado-correctamente');
    FMemoMensaje.Text := '';
    WriteLn( '[Miembro-2]-Mensaje-publicado-desde-interfaz' );
end
else
    MostrarMensaje( 'Error ',
        'No-se-pudo-publicar-el-mensaje.-Verifique-que-la-comunidad-exista' );
end;

```

## 5. Integración de la Estructura al Proyecto

### 5.1. Modificaciones en EstructurasDatos.pas

Listing 9: Clase TEDDMailSystem modificada para Fase 2

```

TEDDMailSystem = class
private
    // Estructuras de Fase 1
    FUsuarios: PUsuario;
    FMatrizFilas: PMatrizDispersaFila;
    FMatrizColumnas: PMatrizDispersaColumna;

    // NUEVAS ESTRUCTURAS DE FASE 2
    FarbolComunidades: PNodeBST;    // rbol BST de comunidades
    FUsuarioActual: PUsuario;

public
    constructor Create;
    destructor Destroy; override;

    // NUEVAS FUNCIONES DE FASE 2 – BST
    function InsertarComunidadBST(var nodo: PNodeBST;
        nombreComunidad, fechaCreacion: String): Boolean;
    function BuscarComunidadBST(nodo: PNodeBST;
        nombre: String): PNodeBST;
    function PublicarMensajeComunidad(nombreComunidad,
        correoUsuario, mensaje: String): Boolean;
    function ListarMensajesComunidad(nombreComunidad: String): String;
    procedure RecorridoInOrdenBST(nodo: PNodeBST);
    procedure GenerarReporteBST(RutaCarpeta: String);
end;

```

## 5.2. Inicialización de la Estructura

Listing 10: Constructor y destructor con BST

```
constructor TEDDMailSystem.Create;
begin
    inherited Create;
    FUsuarios := nil;
    FMatrizFilas := nil;
    FMatrizColumnas := nil;
    FArbolComunidades := nil; // Inicializar rbol BST vac o
    FUsuarioActual := nil;

    // Crear usuario root por defecto
    RegistrarUsuario('Root-Admin', 'root',
        'root@edd.com', '00000000', 'root123', 0);

    WriteLn('[ Sistema]- rbol -BST-de-comunidades-inicializado');
end;

destructor TEDDMailSystem.Destroy;
begin
    // Liberar memoria del BST
    LiberarArbolBST(FArbolComunidades);

    // Liberar otras estructuras...
    inherited Destroy;
end;

procedure TEDDMailSystem.LiberarArbolBST(var nodo: PNodeBST);
var
    mensaje, tempMensaje: PMensajeComunidad;
begin
    if nodo = nil then Exit;

    // Recorrido postorden para liberar memoria
    LiberarArbolBST(nodo^.Izquierdo);
    LiberarArbolBST(nodo^.Derecho);

    // Liberar lista de mensajes
    mensaje := nodo^.ListaMensajes;
    while mensaje <> nil do
    begin
        tempMensaje := mensaje;
        mensaje := mensaje^.Siguiente;
        Dispose(tempMensaje);
    end;

    // Liberar nodo
```

```

    Dispose(nodo);
    nodo := nil;
end;

```

## 6. Validaciones Implementadas

### 6.1. Validaciones de Estructura de Datos (Miembro 1)

Listing 11: Validaciones del BST - Miembro 1

```

function TEDDMailSystem.ValidarNombreComunidad(nombre: String): Boolean;
begin
    Result := True;

    // Validaci n 1: No vac o
    if Trim(nombre) = '' then
    begin
        WriteLn(' [ Validaci n -BST] -Error:-Nombre- vac o ');
        Result := False;
        Exit;
    end;

    // Validaci n 2: Longitud m nima
    if Length(Trim(nombre)) < 3 then
    begin
        WriteLn(' [ Validaci n -BST] -Error:-M nimo-3-caracteres ');
        Result := False;
        Exit;
    end;

    // Validaci n 3: Longitud m xima
    if Length(Trim(nombre)) > 50 then
    begin
        WriteLn(' [ Validaci n -BST] -Error:-M ximo-50-caracteres ');
        Result := False;
        Exit;
    end;

    WriteLn(' [ Validaci n -BST] -Nombre- v lido:-', nombre);
end;

function TEDDMailSystem.ValidarMensaje(mensaje: String): Boolean;
begin
    Result := True;

    if Trim(mensaje) = '' then
    begin
        WriteLn(' [ Validaci n -BST] -Error:-Mensaje- vac o ');

```

```

    Result := False;
    Exit;
end;

if Length(Trim(mensaje)) > 500 then
begin
    WriteLn(' [ Validaci n -BST] -Error: -Mensaje -muy -largo ');
    Result := False;
    Exit;
end;

WriteLn(' [ Validaci n -BST] -Mensaje -v lido ');
end;

```

## 7. Reportes Generados

### 7.1. Reporte de Árbol BST con Graphviz

Listing 12: Generación de reporte BST - Trabajo conjunto

```

procedure TEDDMailSystem.GenerarReporteBST (RutaCarpeta: String);
var
    Archivo: TextFile;
    Process: TProcess;

procedure EscribirNodoBST(nodo: PNodoBST);
var
    nombreLimpio: String;
begin
    if nodo = nil then Exit;

    nombreLimpio := StringReplace(nodo^.NombreComunidad, ' - ', ' _ ',
    [rfReplaceAll]);

    // Escribir nodo actual
    WriteLn(Archivo, Format(' - - "%s" - [ label=" %s\nFecha: -%s\nMensajes: -%d" , -
    ' style=filled , - fillcolor=lightblue ]; ',
    [nombreLimpio, nodo^.NombreComunidad,
    nodo^.FechaCreacion, nodo^.NumeroMensajes]));

    // Escribir conexiones
    if nodo^.Izquierdo <> nil then
    begin
        WriteLn(Archivo, Format(' - - "%s" -> - "%s" - [ label=" izq " ]; ',
        [nombreLimpio,
        StringReplace(nodo^.Izquierdo^.NombreComunidad, ' - ', ' _ ',
        [rfReplaceAll])]));
        EscribirNodoBST(nodo^.Izquierdo);
    end;
    end;

```

```

end;

if nodo^.Derecho <> nil then
begin
  WriteLn( Archivo , Format( ' - "%s" -> "%s" - [label="der"] ; ' ,
    [nombreLimpio ,
      StringReplace(nodo^.Derecho^.NombreComunidad , ' - ' , ' _ ' ,
        [rfReplaceAll]) ] ) );
  EscribirNodoBST(nodo^.Derecho);
end;
end;

begin
try
  ForceDirectories(RutaCarpeta);
  AssignFile(Archivo , RutaCarpeta + '/comunidades_bst.dot');
  Rewrite(Archivo);

  WriteLn(Archivo , 'digraph -BST-{');
  WriteLn(Archivo , ' - - label=" rbol -BST-- Comunidades - Fase - 2"; ');
  WriteLn(Archivo , ' - - fontsize=16; ');
  WriteLn(Archivo , ' - - node - [shape=box]; ');

  if FArbolComunidades = nil then
    WriteLn(Archivo , ' - - empty - [label=" rbol - vac o" , - ' +
      ' style=filled , - fillcolor=lightgray]; ')
  else
    EscribirNodoBST(FArbolComunidades);

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

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

```

```

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

```

## 8. Análisis de la Estructura Árbol BST

### 8.1. Complejidades Algorítmicas

Operación	Complejidad	Descripción
Insertar Comunidad	$O(\log n)$ promedio $O(n)$ peor caso	Inserción ordenada alfabéticamente
Buscar Comunidad	$O(\log n)$ promedio $O(n)$ peor caso	Búsqueda binaria en árbol balanceado
Publicar Mensaje	$O(\log n + m)$	Buscar comunidad + agregar a lista de mensajes
Recorrido InOrden	$O(n)$	Visitar todos los nodos en orden alfabético
Eliminar Comunidad	$O(\log n)$ promedio	Buscar y reorganizar árbol
Generar Reporte	$O(n)$	Recorrido completo del árbol

Cuadro 4: Complejidades del Árbol BST ( $n$  = nodos,  $m$  = mensajes promedio)

### 8.2. Ventajas y Desventajas de la Implementación

Ventajas	Desventajas
<b>Búsqueda Eficiente:</b> $O(\log n)$ en promedio vs $O(n)$ en listas	<b>Degeneración:</b> Puede convertirse en lista si inserciones no balanceadas
<b>Ordenamiento Natural:</b> Las comunidades quedan ordenadas alfabéticamente	<b>Sin Auto-balanceo:</b> No es AVL, puede desbalancearse
<b>Recorridos Flexibles:</b> InOrden, PreOrden, PostOrden disponibles	<b>Complejidad de Código:</b> Recursión requiere cuidado con memoria
<b>Escalabilidad:</b> Mejor rendimiento con muchas comunidades	<b>Memoria:</b> Dos punteros por nodo (izq/der)
<b>Inserción Dinámica:</b> Fácil agregar nuevas comunidades	<b>Eliminación Compleja:</b> Reorganización de subárboles

Cuadro 5: Análisis de ventajas y desventajas del BST

### 8.3. Comparación: Lista de Listas (Fase 1) vs BST (Fase 2)

Aspecto	Lista de Listas (Fase 1)	Árbol BST (Fase 2)
Búsqueda	$O(n)$ lineal	$O(\log n)$ promedio
Inserción	$O(n)$ al final	$O(\log n)$ ordenada
Ordenamiento	Sin orden garantizado	Orden alfabético natural
Memoria	Lista principal + listas secundarias	Nodos con 2 punteros
Complejidad	Más simple de implementar	Requiere manejo de recursión
Escalabilidad	Limitada para grandes datos	Mejor con muchas comunidades
Visualización	Estructura plana	Estructura jerárquica

Cuadro 6: Comparación entre estructuras de Fase 1 y Fase 2

## 9. Testing y Validación

### 9.1. Casos de Prueba Implementados

ID	Caso de Prueba	Resultado Esperado	Responsable
TC01	Crear comunidad válida	Inserción en BST exitosa	M1
TC02	Crear comunidad duplicada	Error: Ya existe	M1
TC03	Buscar comunidad existente	Nodo encontrado	M1
TC04	Buscar comunidad inexistente	nil retornado	M1
TC05	Publicar mensaje válido	Mensaje agregado a lista	M1
TC06	Verificar orden alfabético	Recorrido InOrden correcto	M1
TC07	Interfaz crear comunidad	Formulario funcional	M2
TC08	Interfaz publicar mensaje	Validaciones UI correctas	M2
TC09	Generar reporte BST	Archivos .dot y .png creados	M1
TC10	Integración completa	Sistema funciona correctamente	M2

Cuadro 7: Casos de prueba del Árbol BST



## 10. Conclusiones del Trabajo en Grupo

### 10.1. Logros Alcanzados

- **Migración Exitosa:** Se migró de Lista de Listas (Fase 1) a Árbol BST (Fase 2)
- **Mejora de Eficiencia:** Búsquedas  $O(\log n)$  vs  $O(n)$  anterior
- **Ordenamiento Automático:** Las comunidades quedan ordenadas alfabéticamente
- **Funcionalidad de Mensajes:** Sistema de publicación implementado correctamente
- **Integración Completa:** El BST se integró sin problemas con el sistema existente
- **Interfaz Mejorada:** Nueva UI para gestionar el árbol BST

### 10.2. Lecciones Aprendidas

1. **Importancia del Balanceo:** Un BST sin balanceo puede degradarse
2. **Recursión Eficiente:** El manejo de recursión requiere cuidado con la memoria
3. **Testing Exhaustivo:** Probar casos extremos (árbol degenerado, etc.)
4. **Visualización Clara:** Graphviz ayuda a entender la estructura
5. **Integración Gradual:** Migrar de una estructura a otra requiere planificación

### 10.3. Recomendaciones para Futuros Desarrollos

- Implementar auto-balanceo (convertir a AVL) para garantizar  $O(\log n)$
- Agregar función de eliminación de comunidades
- Implementar búsqueda por fecha de creación
- Optimizar visualización para árboles grandes
- Agregar estadísticas por comunidad (usuarios activos, etc.)

## 11. Anexos

### 11.1. Métricas del Proyecto

Métrica	Valor	Observaciones
Líneas de código BST	500	Entre ambos miembros
Funciones implementadas	10	7 por M1, 3 por M2
Profundidad máxima probada	5 niveles	Testing con 31 nodos
Tiempo de búsqueda promedio	$O(\log n)$	Con árbol balanceado
Casos de prueba	10	Todos exitosos
Tiempo de migración	1 semana	De lista a BST
Reuniones de coordinación	6	3 por semana
Commits al repositorio	18	9 por cada miembro

Cuadro 8: Métricas del proyecto Fase 2

### 11.2. Información de Contacto del Grupo

Información	Daniela Azucena Chinchilla López	José Alexander López López
Email	<a href="mailto:chinchillad230@gmail.com">chinchillad230@gmail.com</a>	<a href="mailto:iosealexander40@outlook.com">iosealexander40@outlook.com</a>
GitHub	Azu-bit	JoseArt777
Especialidad	Estructuras de Datos y Algoritmos	Interfaz Gráfica y UX
Contribución Principal	Implementación BST y reportes	Frontend e integración

Cuadro 9: Información de contacto del grupo