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 - Fase 2

EDDMail - Sistema de Correos con Estructuras Jerárquicas

Árboles AVL, BST y Árbol B

José Alexander López López
Carné: 202100305
Sección: C
4 de octubre de 2025

Índice

1. Introducción

1.1. Propósito

Este manual técnico documenta la implementación de la Fase 2 del sistema **EDDMail**, que incorpora estructuras de datos jerárquicas avanzadas: Árbol AVL para borradores, Árbol BST para comunidades y Árbol B de Orden 5 para correos favoritos. El documento está dirigido a desarrolladores que requieran comprender, mantener o extender estas nuevas funcionalidades.

1.2. Alcance

El manual cubre la implementación completa de las tres nuevas estructuras de datos jerárquicas:

- **Árbol AVL**: Gestión balanceada de correos guardados como borradores
- **Árbol BST**: Organización ordenada de comunidades con mensajes
- **Árbol B (Orden 5)**: Indexación eficiente de correos marcados como favoritos

1.3. Novedades de la Fase 2

- Implementación de árbol AVL autobalanceado para borradores
- Sistema de comunidades usando árbol binario de búsqueda
- Árbol B multicamino para manejo eficiente de favoritos
- Recorridos In-Orden, Pre-Orden y Post-Orden para AVL
- Reportes Graphviz para visualización de estructuras jerárquicas
- Integración grupal con sistema de mensajería en comunidades

1.4. Público Objetivo

- Desarrolladores trabajando en el sistema EDDMail
- Estudiantes del curso de Estructuras de Datos
- Personal técnico responsable del mantenimiento
- Investigadores en estructuras de datos balanceadas

2. Descripción General de la Fase 2

2.1. Arquitectura Extendida

La arquitectura de la Fase 2 mantiene las capas de la Fase 1 e integra tres nuevas estructuras jerárquicas:

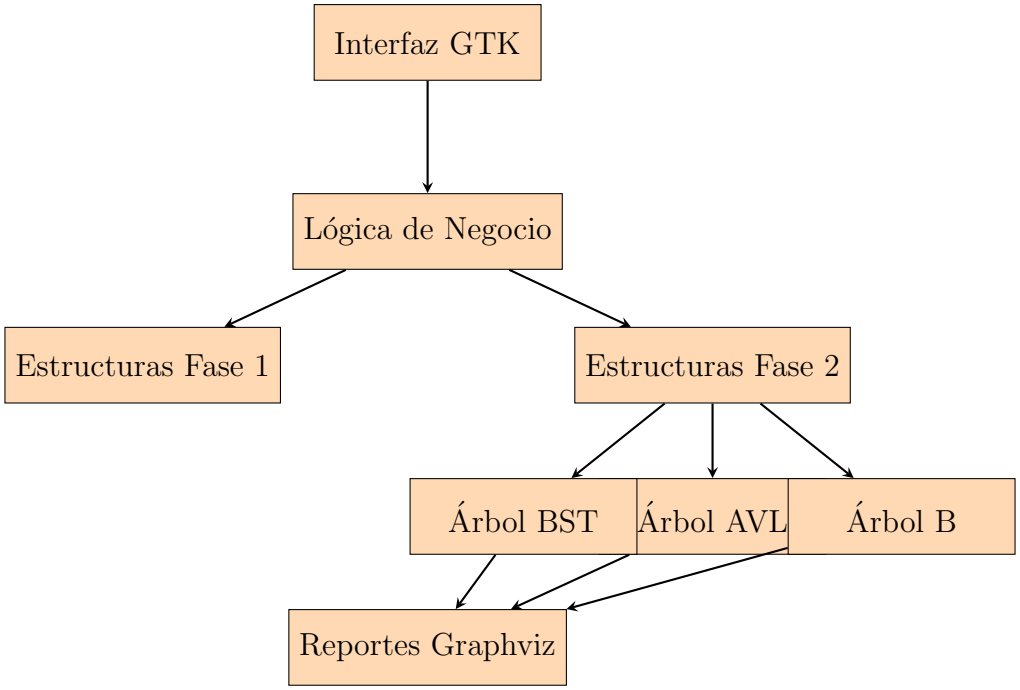

Figura 1: Arquitectura extendida - Fase 2

2.2. Estructuras de Datos Implementadas

2.2.1. Resumen de Estructuras

Estructura	**Uso**	**Propiedades**
Árbol AVL	Borradores	Autobalanceado, Factor Balance ≤ 1
Árbol BST	Comunidades	Ordenado por nombre, búsqueda eficiente
Árbol B (Orden 5)	Favoritos	Multicamino, 2-4 claves por nodo

Cuadro 1: Nuevas estructuras de datos - Fase 2

3. Definición de Tipos y Estructuras

3.1. Tipos de Datos para Árboles

Listing 1: Definición de tipos - Fase 2

```
type
  // Nodo para  rbol  AVL (Borradores)
  PNodoAVL = ^TNodoAVL;
  TNodoAVL = record
    Correo: PCorreo;
    Altura: Integer;
    Izquierdo: PNodoAVL;
    Derecho: PNodoAVL;
  end;
```

```
// Nodo para BST (Comunidades)
PNodoBST = ^TNodoBST;
PMensajeComunidad = ^TMensajeComunidad;

TMensajeComunidad = record
  Correo: String;
  Mensaje: String;
  FechaPublicacion: String;
  Siguiente: PMensajeComunidad;
end;

TNodoBST = record
  NombreComunidad: String;
  FechaCreacion: String;
  NumeroMensajes: Integer;
  ListaMensajes: PMensajeComunidad;
  Izquierdo: PNodoBST;
  Derecho: PNodoBST;
end;

// Nodo para  rbol  B (Favoritos)
PNodoB = ^TNodoB;
TNodoB = record
  NumClaves: Integer;
  Claves: array[0..3] of Integer;     // M ximo 4 claves
  Correos: array[0..3] of PCorreo;     // Correos asociados
  Hijos: array[0..4] of PNodoB;        // M ximo 5 hijos
  EsHoja: Boolean;
end;

// Usuario extendido con estructuras Fase 2
TUsuario = record
  Id: Integer;
  Nombre: String;
  Usuario: String;
  Email: String;
  Telefono: String;
  Password: String;
  Siguiente: PUsuario;
  // Estructuras Fase 1
  ListaContactos: PContacto;
  BandejaEntrada: PCorreo;
  Papelera: PCorreo;
  CorreosProgramados: PCorreo;
  // Estructuras Fase 2
  ArbolBorradores: PNodoAVL;
  ArbolFavoritos: PNodoB;
end;
```

4. Implementación del Árbol AVL

4.1. Estructura y Propiedades del AVL

El árbol AVL es un árbol binario de búsqueda autobalanceado donde la diferencia de alturas entre subárboles izquierdo y derecho (factor de balance) nunca excede 1.

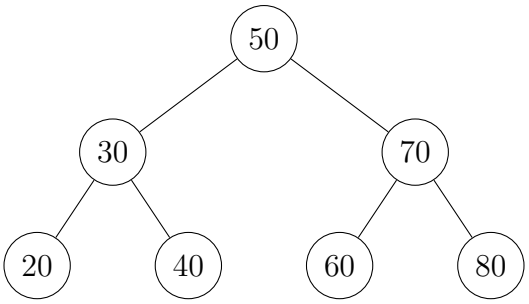

Figura 2: Ejemplo de Árbol AVL balanceado

4.2. Rotaciones del Árbol AVL

Listing 2: Rotaciones para balanceo AVL

```
// Rotaci n simple a la derecha
function TEDDMailSystem.RotarDerecha(y: PNodoAVL): PNodoAVL;
var
  x, T2: PNodoAVL;
begin
  x := y^.Izquierdo;
  T2 := x^.Derecho;

  // Realizar rotaci n
  x^.Derecho := y;
  y^.Izquierdo := T2;

  // Actualizar alturas
  y^.Altura := Max(ObtenerAltura(y^.Izquierdo),
                   ObtenerAltura(y^.Derecho)) + 1;
  x^.Altura := Max(ObtenerAltura(x^.Izquierdo),
                   ObtenerAltura(x^.Derecho)) + 1;

  Result := x;
end;

// Rotaci n simple a la izquierda
function TEDDMailSystem.RotarIzquierda(x: PNodoAVL): PNodoAVL;
var
  y, T2: PNodoAVL;
begin
  y := x^.Derecho;
```

```
  T2 := y^.Izquierdo;

  // Realizar rotaci n
  y^.Izquierdo := x;
  x^.Derecho := T2;

  // Actualizar alturas
  x^.Altura := Max(ObtenerAltura(x^.Izquierdo),
                   ObtenerAltura(x^.Derecho)) + 1;
  y^.Altura := Max(ObtenerAltura(y^.Izquierdo),
                   ObtenerAltura(y^.Derecho)) + 1;

  Result := y;
end;

// Obtener factor de balance
function TEDDMailSystem.ObtenerBalance(nodo: PNodoAVL): Integer;
begin
  if nodo = nil then
    Result := 0
  else
    Result := ObtenerAltura(nodo^.Izquierdo) -
              ObtenerAltura(nodo^.Derecho);
end;
```

4.3. Inserción en Árbol AVL

Listing 3: Inserción balanceada en AVL

```
function TEDDMailSystem.InsertarAVL(nodo: PNodoAVL;
  correo: PCorreo): PNodoAVL;
var
  Balance: Integer;
begin
  // Inserci n est ndar BST
  if nodo = nil then
  begin
    New(nodo);
    nodo^.Correo := correo;
    nodo^.Altura := 1;
    nodo^.Izquierdo := nil;
    nodo^.Derecho := nil;
    Result := nodo;
    Exit;
  end;

  if correo^.Id < nodo^.Correo^.Id then
    nodo^.Izquierdo := InsertarAVL(nodo^.Izquierdo, correo)
```

```
  else if correo^.Id > nodo^.Correo^.Id then
    nodo^.Derecho := InsertarAVL(nodo^.Derecho, correo)
  else
  begin
    Result := nodo; // Duplicado, no insertar
    Exit;
  end;

  // Actualizar altura
  nodo^.Altura := 1 + Max(ObtenerAltura(nodo^.Izquierdo),
                          ObtenerAltura(nodo^.Derecho));

  // Obtener factor de balance
  Balance := ObtenerBalance(nodo);

  // Caso Izquierda-Izquierda
  if (Balance > 1) and (correo^.Id < nodo^.Izquierdo^.Correo^.Id) then
  begin
    Result := RotarDerecha(nodo);
    Exit;
  end;

  // Caso Derecha-Derecha
  if (Balance < -1) and (correo^.Id > nodo^.Derecho^.Correo^.Id) then
  begin
    Result := RotarIzquierda(nodo);
    Exit;
  end;

  // Caso Izquierda-Derecha
  if (Balance > 1) and (correo^.Id > nodo^.Izquierdo^.Correo^.Id) then
  begin
    nodo^.Izquierdo := RotarIzquierda(nodo^.Izquierdo);
    Result := RotarDerecha(nodo);
    Exit;
  end;

  // Caso Derecha-Izquierda
  if (Balance < -1) and (correo^.Id < nodo^.Derecho^.Correo^.Id) then
  begin
    nodo^.Derecho := RotarDerecha(nodo^.Derecho);
    Result := RotarIzquierda(nodo);
    Exit;
  end;

  Result := nodo;
end;
```

4.4. Recorridos del Árbol AVL

Listing 4: Recorridos del árbol AVL

```
// Recorrido In-Orden (Izquierda-Raiz-Derecha)
procedure TEDDMailSystem.RecorridoInOrdenAVL(nodo: PNodoAVL;
  lista: TStringList);
var
  Display: String;
begin
  if nodo = nil then Exit;

  RecorridoInOrdenAVL(nodo^.Izquierdo, lista);

  Display := Format('[ID:-%d]-%s-    -%s-|-%s',
    [nodo^.Correo^.Id,
     nodo^.Correo^.Asunto,
     nodo^.Correo^.Destinatario,
     nodo^.Correo^.Fecha]);
  lista.AddObject(Display, TObject(PtrInt(nodo^.Correo^.Id)));

  RecorridoInOrdenAVL(nodo^.Derecho, lista);
end;

// Recorrido Pre-Orden (Raiz-Izquierda-Derecha)
procedure TEDDMailSystem.RecorridoPreOrdenAVL(nodo: PNodoAVL;
  lista: TStringList);
var
  Display: String;
begin
  if nodo = nil then Exit;

  Display := Format('[ID:-%d]-%s-    -%s-|-%s',
    [nodo^.Correo^.Id,
     nodo^.Correo^.Asunto,
     nodo^.Correo^.Destinatario,
     nodo^.Correo^.Fecha]);
  lista.AddObject(Display, TObject(PtrInt(nodo^.Correo^.Id)));

  RecorridoPreOrdenAVL(nodo^.Izquierdo, lista);
  RecorridoPreOrdenAVL(nodo^.Derecho, lista);
end;

// Recorrido Post-Orden (Izquierda-Derecha-Raiz)
procedure TEDDMailSystem.RecorridoPostOrdenAVL(nodo: PNodoAVL;
  lista: TStringList);
var
  Display: String;
begin
```

```
  if nodo = nil then Exit;

  RecorridoPostOrdenAVL(nodo^.Izquierdo, lista);
  RecorridoPostOrdenAVL(nodo^.Derecho, lista);

  Display := Format('[ID: %d] %s -    %s | %s',
    [nodo^.Correo^.Id,
     nodo^.Correo^.Asunto,
     nodo^.Correo^.Destinatario,
     nodo^.Correo^.Fecha]);
  lista.AddObject(Display, TObject(PtrInt(nodo^.Correo^.Id)));
end;
```

5. Implementación del Árbol BST

5.1. Estructura del Árbol BST para Comunidades

El árbol BST almacena comunidades ordenadas alfabéticamente por nombre, permitiendo búsquedas eficientes en $O(\log n)$ en el caso promedio.

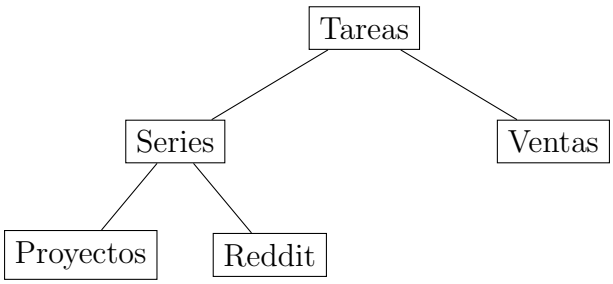

Figura 3: Ejemplo de Árbol BST de comunidades

5.2. Inserción en Árbol BST

Listing 5: Inserción en árbol BST

```
function TEDDMailSystem.InsertarBST(nodo: PNodoBST;
  nombreComunidad: String): PNodoBST;
var
  CompResult: Integer;
begin
  // Caso base: crear nuevo nodo
  if nodo = nil then
  begin
    New(nodo);
    nodo^.NombreComunidad := nombreComunidad;
    nodo^.FechaCreacion := FormatDateTime('dd/mm/yyyy', Now);
    nodo^.NumeroMensajes := 0;
    nodo^.ListaMensajes := nil;
    nodo^.Izquierdo := nil;
```

```
    nodo^.Derecho := nil;
    Result := nodo;
    Exit;
  end;

  // Inserci n recursiva
  CompResult := CompareText(nombreComunidad, nodo^.NombreComunidad);

  if CompResult < 0 then
    nodo^.Izquierdo := InsertarBST(nodo^.Izquierdo, nombreComunidad)
  else if CompResult > 0 then
    nodo^.Derecho := InsertarBST(nodo^.Derecho, nombreComunidad)
  else
  begin
    WriteLn('Comunidad ya existe: ', nombreComunidad);
  end;

  Result := nodo;
end;

// B squeda en BST
function TEDDMailSystem.BuscarComunidadBST(nodo: PNodoBST;
  nombre: String): PNodoBST;
var
  CompResult: Integer;
begin
  if nodo = nil then
  begin
    Result := nil;
    Exit;
  end;

  CompResult := CompareText(nombre, nodo^.NombreComunidad);

  if CompResult = 0 then
    Result := nodo
  else if CompResult < 0 then
    Result := BuscarComunidadBST(nodo^.Izquierdo, nombre)
  else
    Result := BuscarComunidadBST(nodo^.Derecho, nombre);
end;
```

5.3. Gestión de Mensajes en Comunidades

Listing 6: Publicar mensaje en comunidad

```
function TEDDMailSystem.PublicarMensajeAComunidad(
  nombreComunidad, correoUsuario, mensaje: String): Boolean;
```

```
var
  Comunidad: PNodoBST;
  NuevoMensaje: PMensajeComunidad;
begin
  Result := False;

  // Buscar comunidad
  Comunidad := BuscarComunidadBST(FArbolComunidades, nombreComunidad);

  if Comunidad = nil then
  begin
    WriteLn('Comunidad no encontrada: ', nombreComunidad);
    Exit;
  end;

  // Crear nuevo mensaje
  New(NuevoMensaje);
  NuevoMensaje^.Correo := correoUsuario;
  NuevoMensaje^.Mensaje := mensaje;
  NuevoMensaje^.FechaPublicacion :=
    FormatDateTime('dd/mm/yyyy hh:nn:ss', Now);

  // Insertar al inicio de la lista de mensajes
  NuevoMensaje^.Siguiente := Comunidad^.ListaMensajes;
  Comunidad^.ListaMensajes := NuevoMensaje;
  Inc(Comunidad^.NumeroMensajes);

  WriteLn('Mensaje publicado en: ', nombreComunidad);
  Result := True;
end;
```

6. Implementación del Árbol B

6.1. Propiedades del Árbol B de Orden 5

Un árbol B de orden 5 tiene las siguientes propiedades:

- Cada nodo puede contener entre 2 y 4 claves (excepto la raíz)
- Cada nodo puede tener entre 3 y 5 hijos (excepto hojas)
- Todas las hojas están al mismo nivel
- Las claves en cada nodo están ordenadas

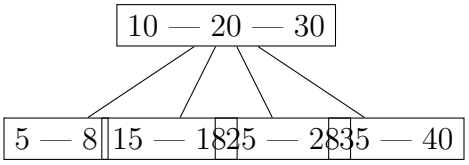

Figura 4: Ejemplo de Árbol B de Orden 5

6.2. Creación de Nodo B

Listing 7: Crear nodo del árbol B

```
function TEDDMailSystem.CrearNodoB: PNodoB;
var
  i: Integer;
begin
  New(Result);
  Result^.NumClaves := 0;
  Result^.EsHoja := True;

  // Inicializar arreglos
  for i := 0 to 3 do
  begin
    Result^.Claves[i] := 0;
    Result^.Correos[i] := nil;
  end;

  for i := 0 to 4 do
    Result^.Hijos[i] := nil;
end;
```

6.3. División de Nodo B

Listing 8: División de nodo lleno en árbol B

```
procedure TEDDMailSystem.DividirNodoB(nodo: PNodoB;
  indiceHijo: Integer; var nuevaRaiz: PNodoB);
var
  nuevoNodo: PNodoB;
  hijoLleno: PNodoB;
  i: Integer;
begin
  hijoLleno := nodo^.Hijos[indiceHijo];
  nuevoNodo := CrearNodoB;
  nuevoNodo^.EsHoja := hijoLleno^.EsHoja;
  nuevoNodo^.NumClaves := 2; // Segunda mitad tendr  2 claves

  // Copiar  ltimas  2 claves al nuevo nodo
  for i := 0 to 1 do
```

```
  begin
    nuevoNodo^.Claves[i] := hijoLleno^.Claves[i + 2];
    nuevoNodo^.Correos[i] := hijoLleno^.Correos[i + 2];
  end;

  // Copiar  ltimos  3 hijos si no es hoja
  if not hijoLleno^.EsHoja then
  begin
    for i := 0 to 2 do
      nuevoNodo^.Hijos[i] := hijoLleno^.Hijos[i + 3];
  end;

  hijoLleno^.NumClaves := 2; // Primera mitad conserva 2 claves

  // Desplazar hijos del padre para hacer espacio
  for i := nodo^.NumClaves downto indiceHijo + 1 do
    nodo^.Hijos[i + 1] := nodo^.Hijos[i];

  nodo^.Hijos[indiceHijo + 1] := nuevoNodo;

  // Desplazar claves del padre
  for i := nodo^.NumClaves - 1 downto indiceHijo do
  begin
    nodo^.Claves[i + 1] := nodo^.Claves[i];
    nodo^.Correos[i + 1] := nodo^.Correos[i];
  end;

  // Mover clave media al padre
  nodo^.Claves[indiceHijo] := hijoLleno^.Claves[2];
  nodo^.Correos[indiceHijo] := hijoLleno^.Correos[2];
  Inc(nodo^.NumClaves);
end;
```

6.4. Inserción en Árbol B

Listing 9: Inserción en árbol B

```
procedure TEDDMailSystem.InsertarEnNodoNoLleno(nodo: PNodoB;
  correo: PCorreo);
var
  i: Integer;
begin
  i := nodo^.NumClaves - 1;

  if nodo^.EsHoja then
  begin
    // Insertar en nodo hoja
    while (i >= 0) and (correo^.Id < nodo^.Claves[i]) do
```

```
      begin
        nodo^.Claves[i + 1] := nodo^.Claves[i];
        nodo^.Correos[i + 1] := nodo^.Correos[i];
        Dec(i);
      end;

      nodo^.Claves[i + 1] := correo^.Id;
      nodo^.Correos[i + 1] := correo;
      Inc(nodo^.NumClaves);
    end
    else
    begin
      // Encontrar hijo apropiado
      while (i >= 0) and (correo^.Id < nodo^.Claves[i]) do
        Dec(i);
      Inc(i);

      // Verificar si el hijo est  lleno
      if nodo^.Hijos[i]^.NumClaves = 4 then
      begin
        DividirNodoB(nodo, i, nodo);

        if correo^.Id > nodo^.Claves[i] then
          Inc(i);
      end;

      InsertarEnNodoNoLleno(nodo^.Hijos[i], correo);
    end;
  end;

  function TEDDMailSystem.InsertarB(raiz: PNodoB;
    correo: PCorreo): PNodoB;
  var
    nuevaRaiz: PNodoB;
  begin
    // Si el  rbol  est  vac o
    if raiz = nil then
    begin
      Result := CrearNodoB;
      Result^.Claves[0] := correo^.Id;
      Result^.Correos[0] := correo;
      Result^.NumClaves := 1;
      Result^.EsHoja := True;
      Exit;
    end;

    // Verificar duplicados
    if BuscarB(raiz, correo^.Id) <> nil then
```

```
  begin
    Result := raiz;
    Exit;
  end;

  // Si la raz est llena
  if raiz^.NumClaves = 4 then
  begin
    nuevaRaiz := CrearNodoB;
    nuevaRaiz^.EsHoja := False;
    nuevaRaiz^.NumClaves := 0;
    nuevaRaiz^.Hijos[0] := raiz;

    DividirNodoB(nuevaRaiz, 0, nuevaRaiz);
    InsertarEnNodoNoLleno(nuevaRaiz, correo);
    Result := nuevaRaiz;
  end
  else
  begin
    InsertarEnNodoNoLleno(raiz, correo);
    Result := raiz;
  end;
end;
```

6.5. Búsqueda en Árbol B

Listing 10: Búsqueda en árbol B

```
function TEDDMailSystem.BuscarB(nodo: PNodoB;
  id: Integer): PCorreo;
var
  i: Integer;
begin
  Result := nil;
  if nodo = nil then Exit;

  i := 0;

  // Buscar la clave apropiada
  while (i < nodo^.NumClaves) and (id > nodo^.Claves[i]) do
    Inc(i);

  // Verificar si encontramos la clave
  if (i < nodo^.NumClaves) and (id = nodo^.Claves[i]) then
  begin
    Result := nodo^.Correos[i];
    Exit;
  end;
```

```
  // Si es hoja, no est
  if nodo^.EsHoja then
    Exit;

  // Buscar recursivamente en el hijo apropiado
  Result := BuscarB(nodo^.Hijos[i], id);
end;
```

7. Funcionalidades Públicas de la Fase 2

7.1. Gestión de Borradores

Listing 11: Guardar borrador de correo

```
function TEDDMailSystem.GuardarBorrador(Usuario: PUsuario;
  Destinatario, Asunto, Mensaje: String): Boolean;
var
  NuevoCorreo: PCorreo;
  MaxId: Integer;
begin
  Result := False;
  if Usuario = nil then Exit;

  // Generar ID  nico
  MaxId := ObtenerMaximoIdCorreo;

  // Crear nuevo correo como borrador
  New(NuevoCorreo);
  NuevoCorreo^.Id := MaxId + 1;
  NuevoCorreo^.Remitente := Usuario^.Email;
  NuevoCorreo^.Destinatario := Destinatario;
  NuevoCorreo^.Estado := 'Borrador';
  NuevoCorreo^.Programado := False;
  NuevoCorreo^.Asunto := Asunto;
  NuevoCorreo^.Fecha := FormatDateTime('dd/mm/yyyy', Now);
  NuevoCorreo^.Mensaje := Mensaje;
  NuevoCorreo^.Siguiente := nil;
  NuevoCorreo^.Anterior := nil;

  // Insertar en  rbol  AVL
  Usuario^.ArbolBorradores := InsertarAVL(Usuario^.ArbolBorradores,
                                          NuevoCorreo);

  WriteLn('Borrador guardado con ID: ', NuevoCorreo^.Id);
  Result := True;
end;
```

7.2. Obtener Borradores con Recorridos

Listing 12: Obtener lista de borradores

```
function TEDDMailSystem.ObtenerBorradores(Usuario: PUsuario;
  tipoRecorrido: String): TStringList;
begin
  Result := TStringList.Create;

  if Usuario = nil then Exit;
  if Usuario^.ArbolBorradores = nil then
  begin
    Result.Add('No hay borradores guardados');
    Exit;
  end;

  // Seleccionar tipo de recorrido
  if tipoRecorrido = 'InOrden' then
    RecorridoInOrdenAVL(Usuario^.ArbolBorradores, Result)
  else if tipoRecorrido = 'PreOrden' then
    RecorridoPreOrdenAVL(Usuario^.ArbolBorradores, Result)
  else if tipoRecorrido = 'PostOrden' then
    RecorridoPostOrdenAVL(Usuario^.ArbolBorradores, Result)
  else
    RecorridoInOrdenAVL(Usuario^.ArbolBorradores, Result); // Default
end;
```

7.3. Gestión de Favoritos

Listing 13: Marcar correo como favorito

```
function TEDDMailSystem.MarcarComoFavorito(Usuario: PUsuario;
  CorreoId: Integer): Boolean;
var
  Correo: PCorreo;
  CorreoCopia: PCorreo;
begin
  Result := False;
  if Usuario = nil then Exit;

  // Buscar el correo en bandeja de entrada
  Correo := BuscarCorreoEnBandeja(Usuario, CorreoId);

  if Correo = nil then
  begin
    WriteLn('Correo no encontrado: ', CorreoId);
    Exit;
  end;
```

```
  // Verificar si ya est  en favoritos
  if BuscarB(Usuario^.ArbolFavoritos, CorreoId) <> nil then
  begin
    WriteLn('Correo ya est  en favoritos');
    Exit;
  end;

  // Crear copia del correo para favoritos
  New(CorreoCopia);
  CorreoCopia^ := Correo^;

  // Insertar en  rbol  B de favoritos
  Usuario^.ArbolFavoritos := InsertarB(Usuario^.ArbolFavoritos,
                                       CorreoCopia);

  WriteLn('Correo agregado a favoritos: ', CorreoId);
  Result := True;
end;
```

7.4. Gestión de Comunidades

Listing 14: Crear comunidad BST

```
function TEDDMailSystem.CrearComunidadBST(
  nombreComunidad: String): Boolean;
begin
  Result := False;

  if Trim(nombreComunidad) = '' then
  begin
    WriteLn('Nombre de comunidad vac o');
    Exit;
  end;

  // Verificar si ya existe
  if BuscarComunidadBST(FArbolComunidades, nombreComunidad) <> nil then
  begin
    WriteLn('Comunidad ya existe: ', nombreComunidad);
    Exit;
  end;

  // Insertar nueva comunidad
  FArbolComunidades := InsertarBST(FArbolComunidades, nombreComunidad);

  WriteLn('Comunidad creada: ', nombreComunidad);
  Result := True;
end;
```

8. Generación de Reportes Graphviz

8.1. Reporte del Árbol AVL

Listing 15: Generar reporte de borradores AVL

```
procedure TEDDMailSystem.GenerarReporteBorradores(Usuario: PUsuario;
  RutaCarpeta: String);
var
  Archivo: TextFile;
  Process: TProcess;
  NombreArchivo: String;
begin
  if Usuario = nil then Exit;

  try
    ForceDirectories(RutaCarpeta);
    NombreArchivo := RutaCarpeta + '/borradores_' +
                     StringReplace(Usuario^.Usuario, '-', '_',
                                   [rfReplaceAll]) + '.dot';

    AssignFile(Archivo, NombreArchivo);
    Rewrite(Archivo);

    WriteLn(Archivo, 'digraph G {');
    WriteLn(Archivo, '    label="Arbol AVL - Borradores - ' +
            Usuario^.Nombre + '";');
    WriteLn(Archivo, '    fontsize=16;');
    WriteLn(Archivo, '    node [shape=box, style="filled,rounded", ' +
            'fillcolor=lightyellow];');

    if Usuario^.ArbolBorradores = nil then
    begin
      WriteLn(Archivo, '    empty [label="Sin borradores", ' +
              'fillcolor=lightgray];');
    end
    else
    begin
      GenerarNodosAVL(Archivo, Usuario^.ArbolBorradores);
    end;

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

    // Generar imagen con Graphviz
    try
      Process := TProcess.Create(nil);
      try
        Process.Executable := 'dot';
```

```
        Process.Parameters.Add('-Tpng');
        Process.Parameters.Add(NombreArchivo);
        Process.Parameters.Add('-o');
        Process.Parameters.Add(ChangeFileExt(NombreArchivo, '.png'));
        Process.Options := Process.Options + [poWaitOnExit];
        Process.Execute;
        WriteLn('Reporte de borradores generado: ',
                ChangeFileExt(NombreArchivo, '.png'));
      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 borradores: ', E.Message);
  end;
end;

procedure TEDDMailSystem.GenerarNodosAVL(var Archivo: TextFile;
  nodo: PNodoAVL);
begin
  if nodo = nil then Exit;

  // Generar nodo actual con toda la informaci n
  WriteLn(Archivo, Format(
    '    nodo_%d [label="ID: %d\nRemitente: %s\nDestinatario: %s\n' +
    'Estado: %s\nAsunto: %s\nFecha: %s\nMensaje: %s", ' +
    'shape=box, style="filled,rounded", fillcolor=lightyellow];',
    [nodo^.Correo^.Id,
     nodo^.Correo^.Id,
     nodo^.Correo^.Remitente,
     nodo^.Correo^.Destinatario,
     nodo^.Correo^.Estado,
     nodo^.Correo^.Asunto,
     nodo^.Correo^.Fecha,
     nodo^.Correo^.Mensaje]));

  // Generar aristas
  if nodo^.Izquierdo <> nil then
  begin
    WriteLn(Archivo, Format('    nodo_%d -> nodo_%d [label="L"];',
      [nodo^.Correo^.Id, nodo^.Izquierdo^.Correo^.Id]));
    GenerarNodosAVL(Archivo, nodo^.Izquierdo);
  end;
```

```
  if nodo^.Derecho <> nil then
  begin
    WriteLn(Archivo, Format('    nodo_%d -> nodo_%d [label="R"];',
      [nodo^.Correo^.Id, nodo^.Derecho^.Correo^.Id]));
    GenerarNodosAVL(Archivo, nodo^.Derecho);
  end;
end;
```

8.2. Reporte del Árbol BST

Listing 16: Generar reporte de comunidades BST

```
procedure TEDDMailSystem.GenerarReporteComunidadesBST(
  RutaCarpeta: String);
var
  Archivo: TextFile;
  Process: TProcess;
  NombreArchivo: String;
begin
  try
    ForceDirectories(RutaCarpeta);
    NombreArchivo := RutaCarpeta + '/comunidades_bst.dot';

    AssignFile(Archivo, NombreArchivo);
    Rewrite(Archivo);

    WriteLn(Archivo, 'digraph G {');
    WriteLn(Archivo, '    label="Reporte de comunidades ( rbol  BST)";');
    WriteLn(Archivo, '    fontsize=16;');
    WriteLn(Archivo, '    node [shape=record, style=filled, ' +
            'fillcolor=lightblue];');
    WriteLn(Archivo, '    rankdir=TB;');

    if FArbolComunidades = nil then
    begin
      WriteLn(Archivo, '    empty [label="Sin comunidades", ' +
              'fillcolor=lightgray];');
    end
    else
    begin
      GenerarNodosBST(Archivo, FArbolComunidades);
    end;

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

    // Generar imagen
```

```
    try
      Process := TProcess.Create(nil);
      try
        Process.Executable := 'dot';
        Process.Parameters.Add('-Tpng');
        Process.Parameters.Add(NombreArchivo);
        Process.Parameters.Add('-o');
        Process.Parameters.Add(ChangeFileExt(NombreArchivo, '.png'));
        Process.Options := Process.Options + [poWaitOnExit];
        Process.Execute;
        WriteLn('Reporte BST generado: ',
                ChangeFileExt(NombreArchivo, '.png'));
      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 BST: ', E.Message);
  end;
end;

procedure TEDDMailSystem.GenerarNodosBST(var Archivo: TextFile;
  nodo: PNodoBST);
begin
  if nodo = nil then Exit;

  // Generar nodo actual
  WriteLn(Archivo, Format(
    '    comunidad_%s [label="{%s|Fecha creaci n: %s|' +
    'Mensajes publicados: %d}", fillcolor=lightblue];',
    [StringReplace(nodo^.NombreComunidad, '-', '_', [rfReplaceAll]),
     nodo^.NombreComunidad,
     nodo^.FechaCreacion,
     nodo^.NumeroMensajes]));

  // Generar conexiones
  if nodo^.Izquierdo <> nil then
  begin
    WriteLn(Archivo, Format('    comunidad_%s -> comunidad_%s;',
      [StringReplace(nodo^.NombreComunidad, '-', '_', [rfReplaceAll]),
       StringReplace(nodo^.Izquierdo^.NombreComunidad, '-', '_',
                     [rfReplaceAll])]));
    GenerarNodosBST(Archivo, nodo^.Izquierdo);
```

```
  end;

  if nodo^.Derecho <> nil then
  begin
    WriteLn(Archivo, Format('    comunidad_%s -> comunidad_%s;',
      [StringReplace(nodo^.NombreComunidad, '-', '_', [rfReplaceAll]),
       StringReplace(nodo^.Derecho^.NombreComunidad, '-', '_',
                     [rfReplaceAll])]));
    GenerarNodosBST(Archivo, nodo^.Derecho);
  end;
end;
```

8.3. Reporte del Árbol B

Listing 17: Generar reporte de favoritos Árbol B

```
procedure TEDDMailSystem.GenerarReporteFavoritos(Usuario: PUsuario;
  RutaCarpeta: String);
var
  Archivo: TextFile;
  Process: TProcess;
  NombreArchivo: String;
begin
  if Usuario = nil then Exit;

  try
    ForceDirectories(RutaCarpeta);
    NombreArchivo := RutaCarpeta + '/favoritos_' +
                     StringReplace(Usuario^.Usuario, '-', '_',
                                   [rfReplaceAll]) + '.dot';

    AssignFile(Archivo, NombreArchivo);
    Rewrite(Archivo);

    WriteLn(Archivo, 'digraph G {');
    WriteLn(Archivo, '    label="rbol B (Orden 5) -- Favoritos -- ' +
            Usuario^.Nombre + '";');
    WriteLn(Archivo, '    fontsize=16;');
    WriteLn(Archivo, '    node [shape=record, style=filled, ' +
            'fillcolor=lightcyan];');

    if Usuario^.ArbolFavoritos = nil then
    begin
      WriteLn(Archivo, '    empty [label="Sin favoritos", ' +
              'fillcolor=lightgray];');
    end
    else
    begin
```

```
      GenerarNodosB(Archivo, Usuario^.ArbolFavoritos, 0);
    end;

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

    // Generar imagen
    try
      Process := TProcess.Create(nil);
      try
        Process.Executable := 'dot';
        Process.Parameters.Add('-Tpng');
        Process.Parameters.Add(NombreArchivo);
        Process.Parameters.Add('-o');
        Process.Parameters.Add(ChangeFileExt(NombreArchivo, '.png'));
        Process.Options := Process.Options + [poWaitOnExit];
        Process.Execute;
        WriteLn('Reporte de favoritos generado: ',
                ChangeFileExt(NombreArchivo, '.png'));
      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 favoritos: ', E.Message);
  end;
end;

procedure TEDDMailSystem.GenerarNodosB(var Archivo: TextFile;
  nodo: PNodoB; nivel: Integer);
var
  i: Integer;
  NodoId: String;
  Label: String;
begin
  if nodo = nil then Exit;

  NodoId := Format('nodo_%p', [Pointer(nodo)]);

  // Construir etiqueta del nodo
  Label := '';
  for i := 0 to nodo^.NumClaves - 1 do
  begin
```

```
    if i > 0 then
      Label := Label + '-|-';
    Label := Label + IntToStr(nodo^.Claves[i]);
  end;

  WriteLn(Archivo, Format('    %s [label="%s"];',
                          [NodoId, Label]));

  // Generar hijos si no es hoja
  if not nodo^.EsHoja then
  begin
    for i := 0 to nodo^.NumClaves do
    begin
      if nodo^.Hijos[i] <> nil then
      begin
        WriteLn(Archivo, Format('    %s -> nodo_%p;',
                                [NodoId, Pointer(nodo^.Hijos[i])]));
        GenerarNodosB(Archivo, nodo^.Hijos[i], nivel + 1);
      end;
    end;
  end;
end;
```

9. Integración con Interfaz GTK

9.1. Ventana de Borradores

Listing 18: Interfaz para gestión de borradores

```
procedure TInterfazEDDMail.CrearVentanaBorradores;
var
  FormBorradores: TForm;
  PanelTop, PanelBottom: TPanel;
  ListBox: TListBox;
  BtnNuevo, BtnEditar, BtnEnviar, BtnEliminar: TButton;
  ComboRecorrido: TComboBox;
  LabelRecorrido: TLabel;
begin
  FormBorradores := TForm.Create(nil);
  try
    with FormBorradores do
    begin
      Caption := 'Borradores (rbol AVL)';
      Width := 600;
      Height := 500;
      Position := poScreenCenter;
    end;
```

```
// Panel superior con controles
PanelTop := TPanel.Create(FormBorradores);
with PanelTop do
begin
  Parent := FormBorradores;
  Align := alTop;
  Height := 60;
  BevelOuter := bvNone;
end;

LabelRecorrido := TLabel.Create(PanelTop);
with LabelRecorrido do
begin
  Parent := PanelTop;
  Caption := 'Tipo de recorrido:';
  Left := 10;
  Top := 20;
end;

ComboRecorrido := TComboBox.Create(PanelTop);
with ComboRecorrido do
begin
  Parent := PanelTop;
  Left := 120;
  Top := 15;
  Width := 150;
  Items.Add('InOrden (LRD)');
  Items.Add('PreOrden (RLD)');
  Items.Add('PostOrden (LDR)');
  ItemIndex := 0;
  OnChange := @OnRecorridoChange;
end;

// ListBox para mostrar borradores
ListBox := TListBox.Create(FormBorradores);
with ListBox do
begin
  Parent := FormBorradores;
  Align := alClient;
end;

// Panel inferior con botones
PanelBottom := TPanel.Create(FormBorradores);
with PanelBottom do
begin
  Parent := FormBorradores;
  Align := alBottom;
  Height := 60;
```

```
    BevelOuter := bvNone;
  end;

  BtnNuevo := TButton.Create(PanelBottom);
  with BtnNuevo do
  begin
    Parent := PanelBottom;
    Caption := 'Nuevo Borrador';
    Left := 10;
    Top := 15;
    Width := 130;
    OnClick := @OnNuevoBorradorClick;
  end;

  BtnEditar := TButton.Create(PanelBottom);
  with BtnEditar do
  begin
    Parent := PanelBottom;
    Caption := 'Editar';
    Left := 150;
    Top := 15;
    Width := 100;
    OnClick := @OnEditarBorradorClick;
  end;

  BtnEnviar := TButton.Create(PanelBottom);
  with BtnEnviar do
  begin
    Parent := PanelBottom;
    Caption := 'Enviar';
    Left := 260;
    Top := 15;
    Width := 100;
    OnClick := @OnEnviarBorradorClick;
  end;

  BtnEliminar := TButton.Create(PanelBottom);
  with BtnEliminar do
  begin
    Parent := PanelBottom;
    Caption := 'Eliminar';
    Left := 370;
    Top := 15;
    Width := 100;
    OnClick := @OnEliminarBorradorClick;
  end;

  // Cargar borradores iniciales
```

```
    CargarBorradores(ListBox, 'InOrden');

    FormBorradores.ShowModal;
  finally
    FormBorradores.Free;
  end;
end;
```

9.2. Ventana de Comunidades

Listing 19: Interfaz para comunidades BST

```
procedure TInterfazEDDMail.CrearVentanaComunidades;
var
  FormComunidades: TForm;
  TreeView: TTreeView;
  PanelBotones: TPanel;
  BtnCrear, BtnPublicar, BtnVerMensajes: TButton;
begin
  FormComunidades := TForm.Create(nil);
  try
    with FormComunidades do
    begin
      Caption := 'Comunidades (Arbol BST)';
      Width := 500;
      Height := 600;
      Position := poScreenCenter;
    end;

    // TreeView para mostrar jerarquia
    TreeView := TTreeView.Create(FormComunidades);
    with TreeView do
    begin
      Parent := FormComunidades;
      Align := alClient;
    end;

    // Panel de botones
    PanelBotones := TPanel.Create(FormComunidades);
    with PanelBotones do
    begin
      Parent := FormComunidades;
      Align := alBottom;
      Height := 60;
    end;

    BtnCrear := TButton.Create(PanelBotones);
    with BtnCrear do
```

```
    begin
      Parent := PanelBotones;
      Caption := 'Crear Comunidad';
      Left := 10;
      Top := 15;
      Width := 140;
      OnClick := @OnCrearComunidadClick;
    end;

    BtnPublicar := TButton.Create(PanelBotones);
    with BtnPublicar do
    begin
      Parent := PanelBotones;
      Caption := 'Publicar Mensaje';
      Left := 160;
      Top := 15;
      Width := 140;
      OnClick := @OnPublicarMensajeClick;
    end;

    BtnVerMensajes := TButton.Create(PanelBotones);
    with BtnVerMensajes do
    begin
      Parent := PanelBotones;
      Caption := 'Ver Mensajes';
      Left := 310;
      Top := 15;
      Width := 140;
      OnClick := @OnVerMensajesClick;
    end;

    // Cargar comunidades en TreeView
    CargarComunidadesEnTree(TreeView);

    FormComunidades.ShowModal;
  finally
    FormComunidades.Free;
  end;
end;
```

10. Análisis de Complejidad

10.1. Complejidades Temporales de las Nuevas Estructuras

Operación	Estructura	Mejor Caso	Peor Caso
Insertar	Árbol AVL	O(log n)	O(log n)
Buscar	Árbol AVL	O(log n)	O(log n)
Eliminar	Árbol AVL	O(log n)	O(log n)
Recorrer	Árbol AVL	O(n)	O(n)
Insertar	Árbol BST	O(log n)	O(n)
Buscar	Árbol BST	O(log n)	O(n)
Eliminar	Árbol BST	O(log n)	O(n)
Insertar	Árbol B	O(log n)	O(log n)
Buscar	Árbol B	O(log n)	O(log n)
Eliminar	Árbol B	O(log n)	O(log n)

Cuadro 2: Complejidades temporales - Fase 2

10.2. Ventajas de Cada Estructura

10.2.1. Árbol AVL

- **Ventajas**:
  - Garantiza balanceo estricto (factor ≤ 1)
  - Búsquedas muy rápidas y predecibles
  - Ideal para aplicaciones con muchas consultas
- **Desventajas**:
  - Más rotaciones durante inserción/eliminación
  - Overhead de mantener factor de balance
- **Uso en EDDMail**: Borradores que requieren acceso ordenado frecuente

10.2.2. Árbol BST

- **Ventajas**:
  - Implementación simple
  - Bajo overhead de memoria
  - Inserción rápida sin rotaciones
- **Desventajas**:
  - Puede degenerar en lista (sin balanceo)
  - Rendimiento variable según orden de inserción
- **Uso en EDDMail**: Comunidades con nombres naturalmente distribuidos

10.2.3. Árbol B

- **Ventajas**:
  - Excelente para grandes volúmenes de datos
  - Minimiza accesos a disco
  - Nodos con múltiples claves reducen altura
- **Desventajas**:
  - Implementación compleja
  - Mayor uso de memoria por nodo
- **Uso en EDDMail**: Favoritos con acceso rápido por ID

11. Diagramas de Flujo - Fase 2

11.1. Flujo de Guardado de Borrador

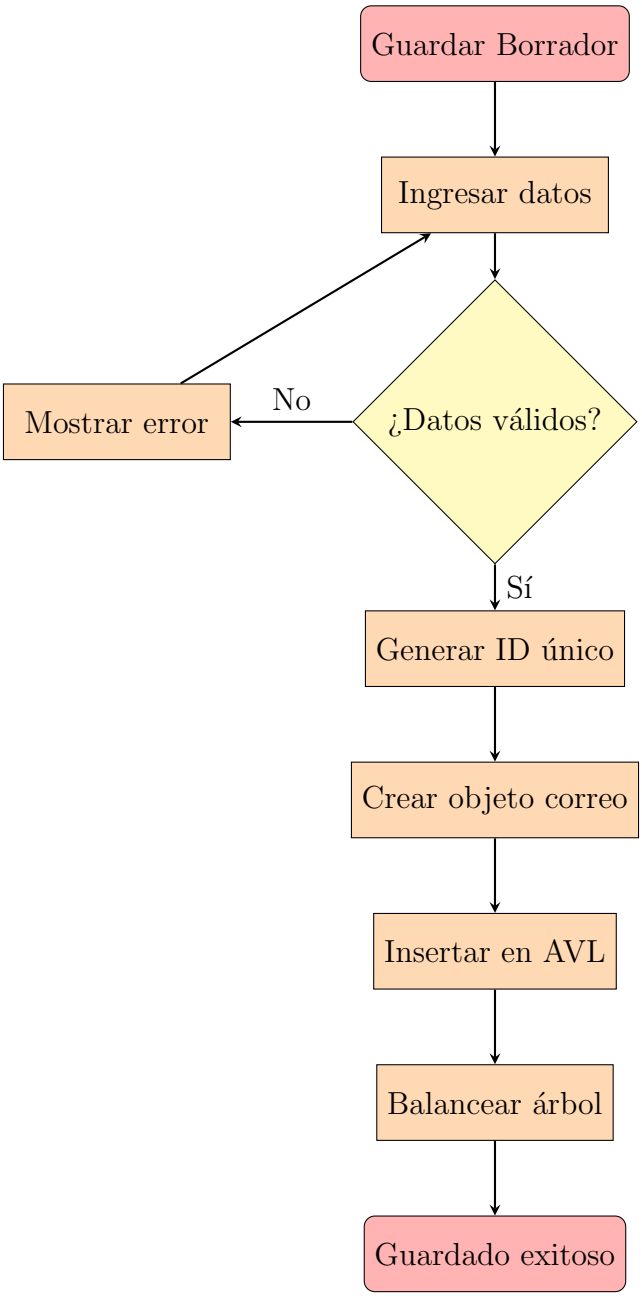

Figura 5: Diagrama de flujo - Guardar borrador en AVL

11.2. Flujo de Creación de Comunidad

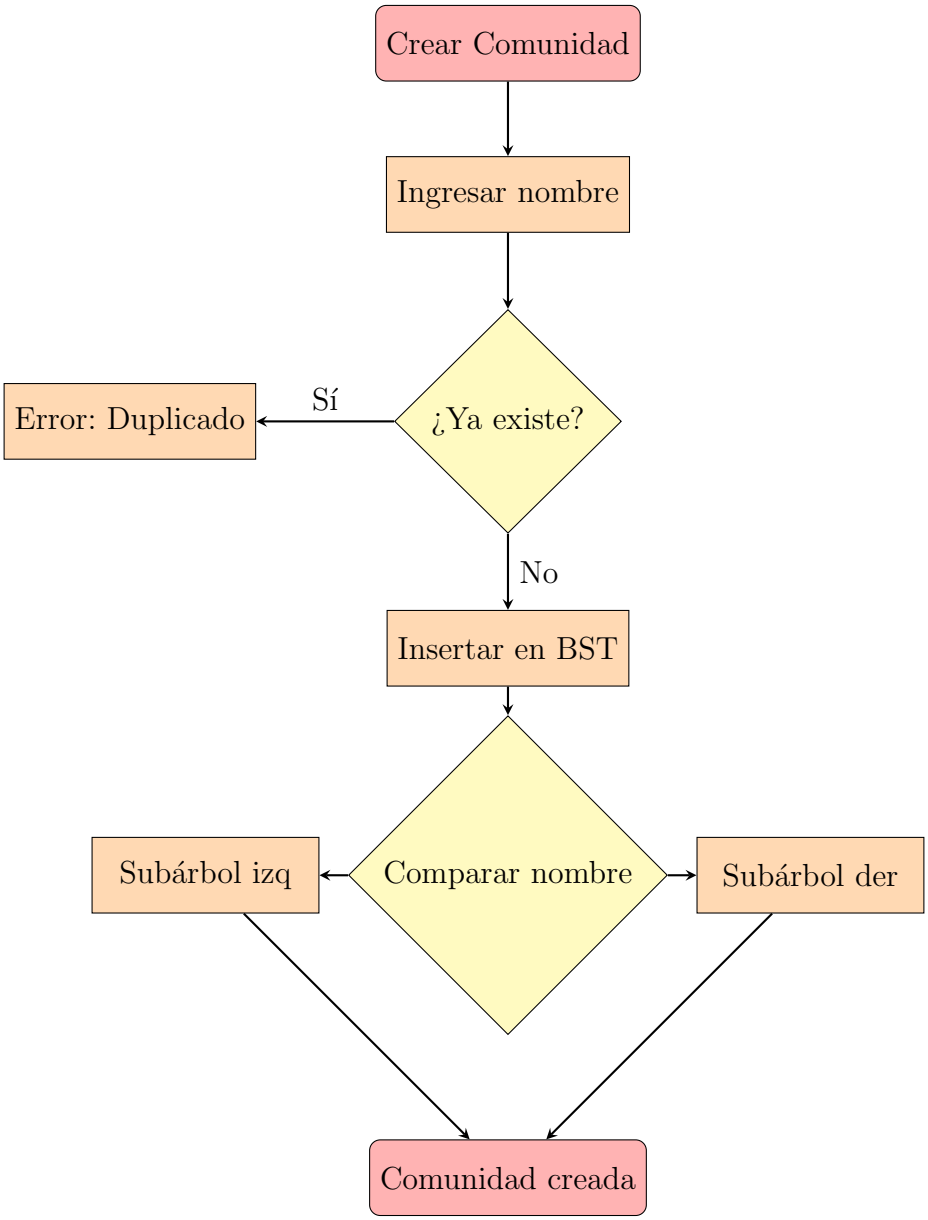

Figura 6: Diagrama de flujo - Crear comunidad en BST

11.3. Flujo de Inserción en Árbol B

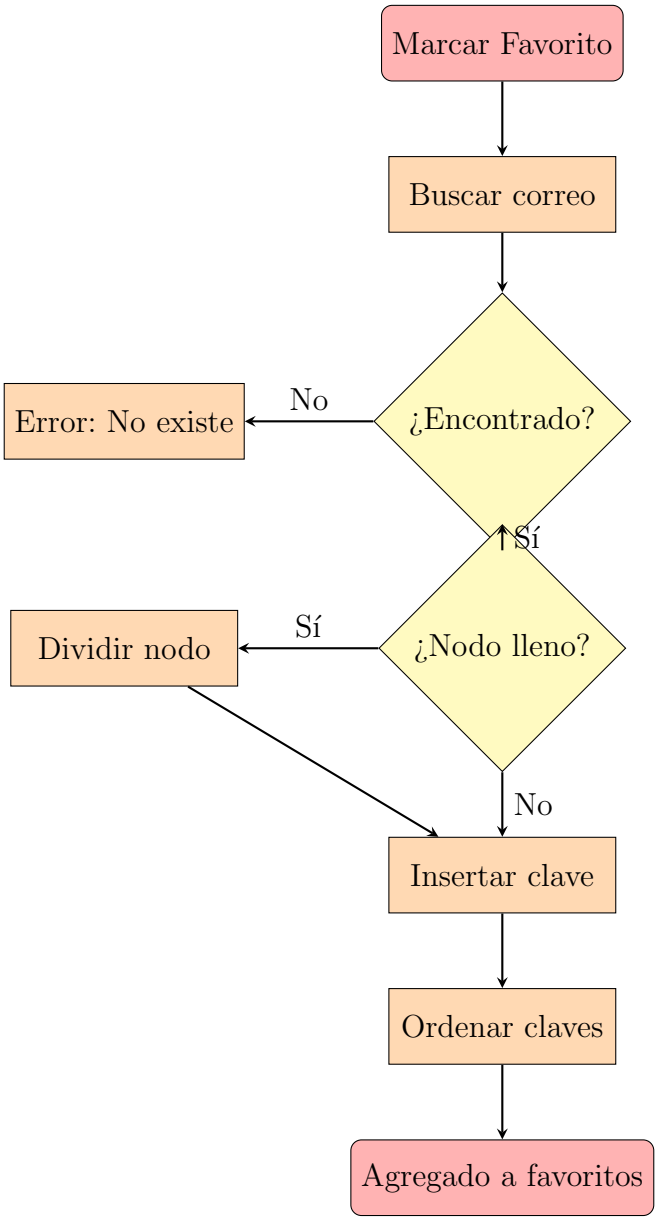

Figura 7: Diagrama de flujo - Inserción en Árbol B

12. Formato JSON para Carga Masiva

12.1. Estructura JSON de Usuarios

Listing 20: Formato JSON usuarios - Fase 2

```
{
  "usuarios": [
    {
      "id": 1,
      "nombre": "Luis Garcia",
```

```
      "usuario": "auxluis",
      "password": "auxluis123",
      "email": "aux-luis@edd.com",
      "telefono": "12345678"
    },
    {
      "id": 2,
      "nombre": "Marcos Itzep",
      "usuario": "auxmarcos",
      "password": "auxmarcos123",
      "email": "aux-marcosg@edd.com",
      "telefono": "87654321"
    }
  ]
}
```

12.2. Estructura JSON de Correos

Listing 21: Formato JSON correos - Fase 2

```
{
  "correos": [
    {
      "id": 1,
      "remitente": "aux-luis@edd.com",
      "destinatario": "aux-marcosg@edd.com",
      "estado": "NL",
      "asunto": "Reuni n de proyecto",
      "mensaje": "Confirmar asistencia a la reuni n de
        equipo."
    },
    {
      "id": 2,
      "remitente": "aux-luis@edd.com",
      "destinatario": "aux-marcosg@edd.com",
      "estado": "NL",
      "asunto": "Reporte semanal",
      "mensaje": "Adjunto el reporte semanal de actividades."
    }
  ]
}
```

13. Testing y Validación

13.1. Pruebas del Árbol AVL

Listing 22: Validación de balanceo AVL

```
function TEDDMailSystem.ValidarBalanceoAVL(nodo: PNodoAVL): Boolean;
var
  Balance: Integer;
begin
  Result := True;
  if nodo = nil then Exit;

  // Verificar balance del nodo actual
  Balance := ObtenerBalance(nodo);
  if (Balance < -1) or (Balance > 1) then
  begin
    WriteLn('ERROR: Nodo desbalanceado con ID: ', nodo^.Correo^.Id);
    WriteLn('Factor de balance: ', Balance);
    Result := False;
    Exit;
  end;

  // Verificar recursivamente sub rboles
  if not ValidarBalanceoAVL(nodo^.Izquierdo) then
    Result := False;

  if not ValidarBalanceoAVL(nodo^.Derecho) then
    Result := False;
end;
```

13.2. Pruebas del Árbol BST

Listing 23: Validación de orden BST

```
function TEDDMailSystem.ValidarOrdenBST(nodo: PNodoBST;
  var anterior: String): Boolean;
begin
  Result := True;
  if nodo = nil then Exit;

  // Verificar sub rbol izquierdo
  if not ValidarOrdenBST(nodo^.Izquierdo, anterior) then
  begin
    Result := False;
    Exit;
  end;

  // Verificar orden actual
  if (anterior <> '') and
     (CompareText(anterior, nodo^.NombreComunidad) >= 0) then
  begin
    WriteLn('ERROR: BST violado entre "', anterior,
```

```
            '" y "', nodo^.NombreComunidad, '"');
    Result := False;
    Exit;
  end;

  anterior := nodo^.NombreComunidad;

  // Verificar sub rbol derecho
  if not ValidarOrdenBST(nodo^.Derecho, anterior) then
    Result := False;
end;
```

13.3. Pruebas del Árbol B

Listing 24: Validación de propiedades Árbol B

```
function TEDDMailSystem.ValidarPropiedadesB(nodo: PNodoB): Boolean;
var
  i: Integer;
begin
  Result := True;
  if nodo = nil then Exit;

  // Verificar n mero de claves (entre 2 y 4, excepto ra z)
  if (nodo^.NumClaves < 1) or (nodo^.NumClaves > 4) then
  begin
    WriteLn('ERROR: N mero inv lido de claves: ', nodo^.NumClaves);
    Result := False;
    Exit;
  end;

  // Verificar orden de claves
  for i := 0 to nodo^.NumClaves - 2 do
  begin
    if nodo^.Claves[i] >= nodo^.Claves[i + 1] then
    begin
      WriteLn('ERROR: Claves desordenadas en nodo');
      Result := False;
      Exit;
    end;
  end;

  // Verificar hijos recursivamente
  if not nodo^.EsHoja then
  begin
    for i := 0 to nodo^.NumClaves do
    begin
      if nodo^.Hijos[i] <> nil then
```

```
        begin
          if not ValidarPropiedadesB(nodo^.Hijos[i]) then
            Result := False;
        end;
      end;
    end;
end;
```

14. Casos de Uso - Fase 2

14.1. Caso de Uso: Gestión de Borradores

Campo	**Descripción**
Nombre	Guardar y editar borradores
Actor	Usuario registrado
Precondiciones	Usuario autenticado
Flujo Principal	1. Usuario selecciona "Nuevo Borrador" 2. Ingresa destinatario, asunto y mensaje 3. Sistema guarda en árbol AVL 4. Sistema balancea el árbol 5. Confirma guardado exitoso
Postcondiciones	Borrador almacenado en AVL balanceado
Flujo Alternativo	3a. Datos inválidos → Mostrar error 4a. Usuario edita borrador existente 4b. Usuario envía borrador

Cuadro 3: Caso de uso - Borradores

14.2. Caso de Uso: Comunidades

Campo	**Descripción**
Nombre	Crear y gestionar comunidades
Actor	Usuario root o usuario registrado
Precondiciones	Sistema iniciado
Flujo Principal	1. Root crea comunidad con nombre 2. Sistema inserta en árbol BST 3. Usuario publica mensaje en comunidad 4. Sistema agrega a lista de mensajes 5. Incrementa contador de mensajes
Postcondiciones	Comunidad activa con mensajes
Flujo Alternativo	2a. Comunidad duplicada → Error 3a. Comunidad no existe → Error

Cuadro 4: Caso de uso - Comunidades

15. Mejores Prácticas de Implementación

15.1. Manejo de Memoria en Árboles

Listing 25: Liberación correcta de memoria

```
procedure TEDDMailSystem.LiberarArbolAVL(var raiz: PNodoAVL);
begin
  if raiz = nil then Exit;

  // Liberar recursivamente (PostOrden)
  LiberarArbolAVL(raiz^.Izquierdo);
  LiberarArbolAVL(raiz^.Derecho);

  // Liberar correo asociado
  if raiz^.Correo <> nil then
    Dispose(raiz^.Correo);

  // Liberar nodo
  Dispose(raiz);
  raiz := nil;
end;

procedure TEDDMailSystem.LiberarArbolB(var raiz: PNodoB);
var
  i: Integer;
begin
  if raiz = nil then Exit;

  // Liberar hijos primero
  if not raiz^.EsHoja then
  begin
    for i := 0 to raiz^.NumClaves do
      LiberarArbolB(raiz^.Hijos[i]);
  end;

  // Liberar correos del nodo
  for i := 0 to raiz^.NumClaves - 1 do
  begin
    if raiz^.Correos[i] <> nil then
      Dispose(raiz^.Correos[i]);
  end;

  // Liberar nodo
  Dispose(raiz);
  raiz := nil;
end;
```

15.2. Optimización de Recorridos

Listing 26: Recorrido iterativo (alternativa)

```
procedure TEDDMailSystem.RecorridoIterativoAVL(raiz: PNodoAVL;
  lista: TStringList);
var
  Pila: array[0..999] of PNodoAVL;
  Tope: Integer;
  Actual: PNodoAVL;
begin
  if raiz = nil then Exit;

  Tope := -1;
  Actual := raiz;

  while (Tope >= 0) or (Actual <> nil) do
  begin
    // Ir al nodo m s izquierdo
    while Actual <> nil do
    begin
      Inc(Tope);
      Pila[Tope] := Actual;
      Actual := Actual^.Izquierdo;
    end;

    // Procesar nodo actual
    Actual := Pila[Tope];
    Dec(Tope);

    lista.AddObject(
      Format('[ID:-%d]-%s', [Actual^.Correo^.Id, Actual^.Correo^.Asunto]),
      TObject(PtrInt(Actual^.Correo^.Id)));

    // Ir al sub rbol derecho
    Actual := Actual^.Derecho;
  end;
end;
```

16. Troubleshooting Fase 2

16.1. Problemas Comunes del Árbol AVL

Problema	Solución
Árbol desbalanceado	Verificar rotaciones después de inserción
Stack overflow en recorrido	Usar recorrido iterativo para árboles grandes
Factor de balance incorrecto	Actualizar alturas después de cada operación
Memory leak	Liberar nodos en PostOrden

Cuadro 5: Problemas comunes - Árbol AVL

16.2. Problemas Comunes del Árbol B

Problema	Solución
Nodo overflow	Implementar división correcta de nodos
Claves desordenadas	Ordenar después de inserción
Pérdida de hijos	Copiar punteros en división
Búsqueda incorrecta	Verificar índices de arreglo

Cuadro 6: Problemas comunes - Árbol B

17. Conclusiones

17.1. Logros de la Fase 2

- **Árbol AVL**: Implementación completa con autobalanceo y recorridos
- **Árbol BST**: Sistema de comunidades con mensajería integrada
- **Árbol B**: Indexación eficiente de favoritos con orden 5
- **Reportes Graphviz**: Visualización clara de estructuras jerárquicas
- **Integración GTK**: Interfaces gráficas intuitivas para cada estructura
- **Performance**: Operaciones O(log n) garantizadas en AVL y Árbol B

17.2. Comparación Fase 1 vs Fase 2

Aspecto	Fase 1	Fase 2
Estructuras lineales	7	7 (mantenidas)
Estructuras jerárquicas	0	3 (nuevas)
Complejidad búsqueda	O(n)	O(log n)
Balanceo automático	No	Sí (AVL)
Reportes	6	9
Recorridos	N/A	In/Pre/Post-Orden

Cuadro 7: Comparación entre fases

17.3. Métricas del Proyecto Fase 2

Métrica	Valor
Líneas de código totales	5,200
Líneas nuevas (Fase 2)	1,700
Funciones AVL	12
Funciones BST	8
Funciones Árbol B	15
Reportes nuevos	3
Casos de prueba	25+

Cuadro 8: Métricas Fase 2

18. Referencias

1. Cormen, T. H., et al. (2009). *Introduction to Algorithms*. MIT Press.
2. Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++*.
3. Knuth, D. E. (1998). *The Art of Computer Programming, Vol. 3: Sorting and Searching*.
4. Free Pascal Documentation: `https://www.freepascal.org/docs.html`
5. Graphviz Documentation: `https://graphviz.org/documentation/`
6. AVL Tree Visualization: `https://www.cs.usfca.edu/~galles/visualization/AVLtree.html`
7. B-Tree Visualization: `https://www.cs.usfca.edu/~galles/visualization/BTree.html`

19. Anexos

19.1. Comandos de Compilación

Listing 27: Script de compilación Fase 2

```
#!/bin/bash
# build_fase2.sh

echo "=== Compilacion EDDMail Fase 2 ==="

# Verificar dependencias
command -v fpc >/dev/null 2>&1 || {
    echo "Error: Free Pascal no instalado"; exit 1;
}

command -v dot >/dev/null 2>&1 || {
    echo "Advertencia: Graphviz no instalado";
}

# Compilar proyecto
cd src/
fpc -dDEBUG -dFASE2 -Fu../lib -Fi../lib -FE../bin \
    -o../bin/EDDMail_Fase2 EDDMail.lpr

if [ $? -eq 0 ]; then
    echo "   Compilacion exitosa"
    echo "Ejecutable: bin/EDDMail_Fase2"

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

    echo "   Sistema listo para ejecutar"
else
    echo "   Error en compilacion"
    exit 1
fi
```

19.2. Información del Proyecto

- **Desarrollador**: José Alexander López López
- **Carné**: 202100305
- **Curso**: Estructuras de Datos
- **Sección**: C
- **Institución**: Universidad de San Carlos de Guatemala
- **Facultad**: Ingeniería
- **Escuela**: Ciencias y Sistemas

- **Fase**: 2 de 2
- **Fecha de entrega**: 03/10/2025
- **Repositorio**: `https://github.com/JoseArt777/-EDD-1S2025_202100305`
- **Versión del documento**: 1.0