

**Link del repositorio:**

[https://github.com/JoseArt777/-EDD-1S2025\\_202100305.git](https://github.com/JoseArt777/-EDD-1S2025_202100305.git)

**Código fuente de la tarea:**

**(Después del código fuente está la gráfica generada)**

```
program Project1;

{$mode objfpc}{$H+}

uses
  Classes, SysUtils, fpjson, jsonparser, Process;

type
  // Estructura del nodo del árbol BST
  PNodeTree = ^TNodeTree;
  TNodeTree = record
    id: Integer;
    first_name: string;
    last_name: string;
    email: string;
    izquierdo: PNodeTree;
    derecho: PNodeTree;
  end;

  // Registro para los datos del JSON
  TPersona = record
    id: Integer;
    first_name: string;
    last_name: string;
    email: string;
  end;

var
  raizBST: PNodeTree;
  personas: array of TPersona;

// Función para insertar en el BST
procedure InsertarEnBST(var nodo: PNodeTree; persona: TPersona);
begin
  if nodo = nil then
  begin
    New(nodo);
    nodo^.id := persona.id;
    nodo^.first_name := persona.first_name;
    nodo^.last_name := persona.last_name;
```

```

    nodo^.email := persona.email;
    nodo^.izquierdo := nil;
    nodo^.derecho := nil;
    WriteLn('Insertado: ID ', persona.id, ' - ', persona.first_name, ' ', persona.last_name);
end
else if persona.id < nodo^.id then
    InsertarEnBST(nodo^.izquierdo, persona)
else if persona.id > nodo^.id then
    InsertarEnBST(nodo^.derecho, persona);
// Si el ID es igual, no insertamos (evitamos duplicados)
end;

// Función para liberar memoria del BST
procedure LiberarBST(nodo: PNodeTree);
begin
    if nodo <> nil then
    begin
        LiberarBST(nodo^.izquierdo);
        LiberarBST(nodo^.derecho);
        Dispose(nodo);
    end;
end;

// Función para recorrer el árbol en orden
procedure RecorrerInOrden(nodo: PNodeTree);
begin
    if nodo <> nil then
    begin
        RecorrerInOrden(nodo^.izquierdo);
        WriteLn('ID: ', nodo^.id, ' | Nombre: ', nodo^.first_name, ' ', nodo^.last_name, ' | Email: ',
nodo^.email);
        RecorrerInOrden(nodo^.derecho);
    end;
end;

// Función para cargar el archivo JSON
function CargarJSON(const nombreArchivo: string): Boolean;
var
    jsonString: string;
    jsonData: TJSONData;
    jsonArray: TJSONArray;
    jsonObject: TJSONObject;
    i: Integer;
    fileStream: TStringList;
begin
    Result := False;
    try
        WriteLn('Cargando archivo JSON: ', nombreArchivo);

```

```

// Leer archivo JSON
fileStream := TStringList.Create;
try
    fileStream.LoadFromFile(nombreArchivo);
    jsonString := fileStream.Text;
finally
    fileStream.Free;
end;

// Parsear JSON
jsonData := GetJSON(jsonString);
if jsonData is TJSONArray then
begin
    jsonArray := TJSONArray(jsonData);
    SetLength(personas, jsonArray.Count);

    WriteLn('Procesando ', jsonArray.Count, ' registros...');

    // Extraer datos de cada persona
    for i := 0 to jsonArray.Count - 1 do
    begin
        jsonObject := TJSONObject(jsonArray[i]);
        personas[i].id := jsonObject.Get('id', 0);
        personas[i].first_name := jsonObject.Get('first_name', '');
        personas[i].last_name := jsonObject.Get('last_name', '');
        personas[i].email := jsonObject.Get('email', '');
    end;

    Result := True;
    WriteLn('JSON cargado exitosamente: ', Length(personas), ' registros');
end;
except
    on E: Exception do
    begin
        WriteLn('Error al cargar JSON: ', E.Message);
    end;
end;

if Assigned(jsonData) then
    jsonData.Free;
end;

// Función para escribir nodos en formato DOT
procedure EscribirNodosGraphviz(nodo: PNodeTree; var archivo: TextFile);
begin
    if nodo <> nil then
    begin

```

```

    WriteLn(archivo, ' ', nodo^.id, ' [label=', nodo^.id, '\n', nodo^.first_name, ' ',
nodo^.last_name, "];");
    EscribirNodosGraphviz(nodo^.izquierdo, archivo);
    EscribirNodosGraphviz(nodo^.derecho, archivo);
end;
end;

```

```

// Función para escribir enlaces en formato DOT
procedure EscribirEnlacesGraphviz(nodo: PNodeTree; var archivo: TextFile);
begin
    if nodo <> nil then
    begin
        if nodo^.izquierdo <> nil then
            WriteLn(archivo, ' ', nodo^.id, ' -> ', nodo^.izquierdo^.id, " ");
        if nodo^.derecho <> nil then
            WriteLn(archivo, ' ', nodo^.id, ' -> ', nodo^.derecho^.id, " ");

        EscribirEnlacesGraphviz(nodo^.izquierdo, archivo);
        EscribirEnlacesGraphviz(nodo^.derecho, archivo);
    end;
end;

```

```

// Función para generar archivo DOT de Graphviz
procedure GenerarGraphviz(const nombreArchivo: string);
var
    archivo: TextFile;
begin
    try
        WriteLn('Generando archivo DOT: ', nombreArchivo);
        AssignFile(archivo, nombreArchivo);
        Rewrite(archivo);

        WriteLn(archivo, 'digraph BST {');
        WriteLn(archivo, ' node [shape=circle, style=filled, fillcolor=lightblue];');
        WriteLn(archivo, ' graph [ordering=out];');
        WriteLn(archivo, ' rankdir=TB;');
        WriteLn(archivo, "");

        // Escribir nodos
        EscribirNodosGraphviz(raizBST, archivo);
        WriteLn(archivo, "");

        // Escribir enlaces
        EscribirEnlacesGraphviz(raizBST, archivo);

        WriteLn(archivo, '}');
        CloseFile(archivo);
    end;
end;

```

```

    WriteLn('Archivo DOT generado exitosamente: ', nombreArchivo);
except
    on E: Exception do
    begin
        WriteLn('Error al generar archivo DOT: ', E.Message);
    end;
end;
end;

// Función para generar imagen PNG usando Graphviz
procedure GenerarImagenPNG(const archivoDOT, archivoPNG: string);
var
    proceso: TProcess;
begin
    try
        WriteLn('Generando imagen PNG: ', archivoPNG);
        proceso := TProcess.Create(nil);
        try
            proceso.Executable := 'dot';
            proceso.Parameters.Add('-Tpng');
            proceso.Parameters.Add(archivoDOT);
            proceso.Parameters.Add('-o');
            proceso.Parameters.Add(archivoPNG);
            proceso.Options := proceso.Options + [poWaitOnExit, poUsePipes];
            proceso.Execute;

            if FileExists(archivoPNG) then
            begin
                WriteLn('Imagen PNG generada exitosamente: ', archivoPNG);
                WriteLn('Puede abrir el archivo PNG para visualizar el árbol BST.');
            end
            else
            begin
                WriteLn('Error: No se pudo generar la imagen PNG.');
                WriteLn('Asegúrese de tener Graphviz instalado: sudo apt install graphviz');
            end;
        finally
            proceso.Free;
        end;
    except
        on E: Exception do
        begin
            WriteLn('Error al ejecutar Graphviz: ', E.Message);
            WriteLn('Instale Graphviz con: sudo apt install graphviz');
        end;
    end;
end;
end;

```

```

// Función para mostrar el menú
procedure MostrarMenu;
begin
    WriteLn;
    WriteLn('=====');
    WriteLn('  BST JSON Loader - Tarea 2  ');
    WriteLn('=====');
    WriteLn('1. Cargar archivo JSON');
    WriteLn('2. Mostrar árbol (recorrido in-orden)');
    WriteLn('3. Generar gráfico con Graphviz');
    WriteLn('4. Salir');
    WriteLn('=====');
    Write('Seleccione una opción: ');
end;

// Programa principal
var
    opcion: Integer;
    nombreArchivo, archivoDOT, archivoPNG: string;
    i: Integer;

begin
    raizBST := nil;
    SetLength(personas, 0);

    WriteLn('BST JSON Loader - Tarea 2');
    WriteLn('Estudiante: [Tu nombre aquí]');
    WriteLn('Carnet: [Tu carnet aquí]');
    WriteLn;

    repeat
        MostrarMenu;
        ReadLn(opcion);

        case opcion of
            1: begin
                Write('Ingrese el nombre del archivo JSON (ej: datos.json): ');
                ReadLn(nombreArchivo);

                if FileExists(nombreArchivo) then
                    begin
                        // Limpiar árbol anterior
                        LiberarBST(raizBST);
                        raizBST := nil;

                        // Cargar JSON
                        if CargarJSON(nombreArchivo) then
                            begin

```

```

    WriteLn;
    WriteLn('Construyendo árbol BST...');
    for i := 0 to Length(personas) - 1 do
    begin
        InsertarEnBST(raizBST, personas[i]);
    end;
    WriteLn;
    WriteLn('Árbol BST construido exitosamente con ', Length(personas), ' nodos.');
```

end;

```

end
else
begin
    WriteLn('Error: El archivo no existe.');
```

end;

```

end;
```

```

2: begin
    if raizBST = nil then
    begin
        WriteLn('Error: Primero debe cargar un archivo JSON.');
```

end

```

    else
    begin
        WriteLn;
        WriteLn('Recorrido In-Orden del árbol BST:');
```

WriteLn('=====');

```

        RecorrerInOrden(raizBST);
        WriteLn('=====');
```

end;

```

end;
```

```

3: begin
    if raizBST = nil then
    begin
        WriteLn('Error: Primero debe cargar un archivo JSON.');
```

end

```

    else
    begin
        archivoDOT := 'bst_tree.dot';
        archivoPNG := 'bst_tree.png';

        WriteLn;
        GenerarGraphviz(archivoDOT);
        GenerarImagenPNG(archivoDOT, archivoPNG);

        WriteLn;
        WriteLn('Archivos generados:');
```

WriteLn('- Código DOT: ', archivoDOT);

```
        WriteLn('- Imagen PNG: ', archivoPNG);
    end;
end;

4: begin
    WriteLn('Liberando memoria...');
    LiberarBST(raizBST);
    WriteLn('¡Hasta luego!');
end;

else
    WriteLn('Opción inválida. Intente de nuevo.');
```

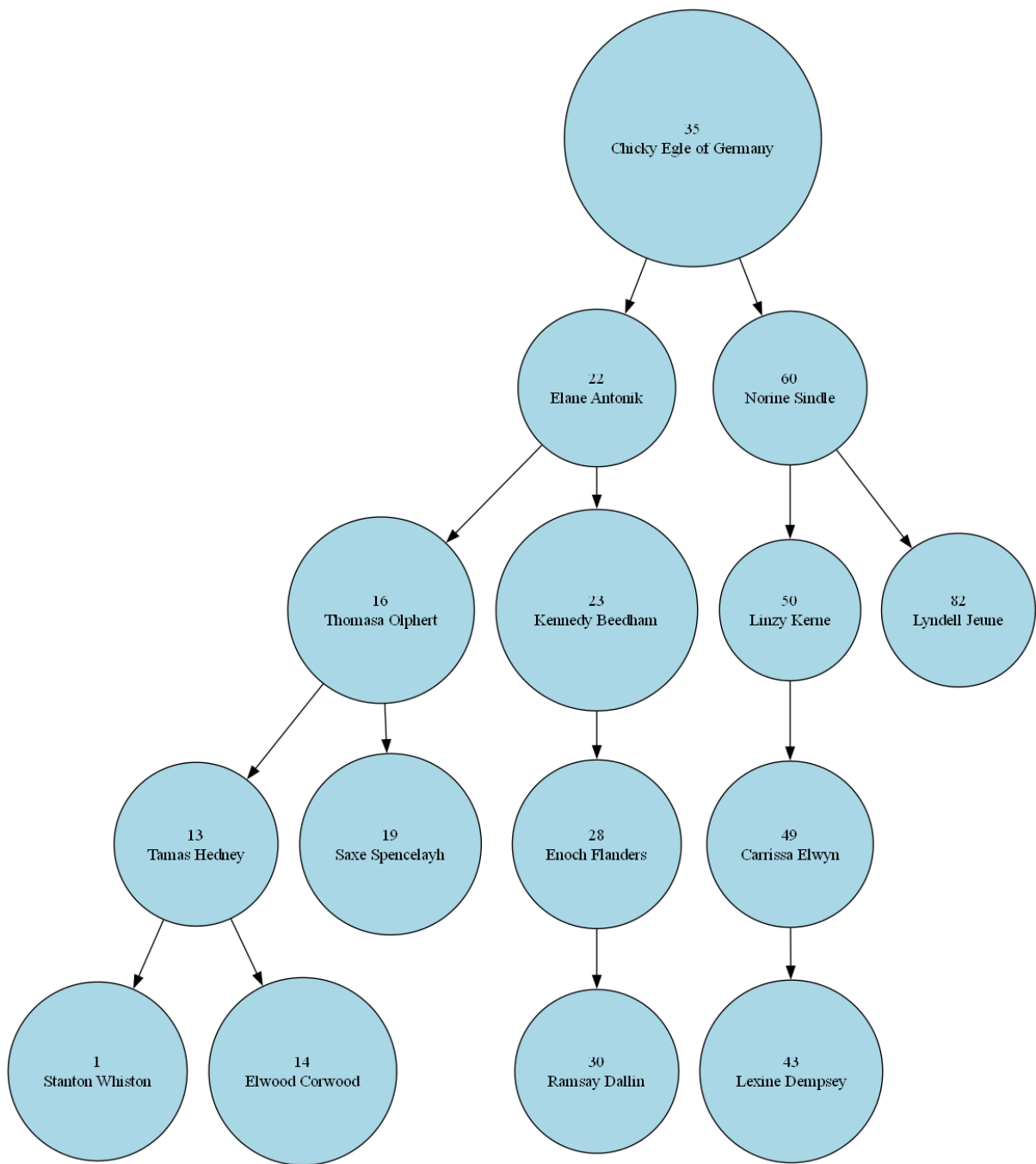
end;

```
if opcion <> 4 then
begin
    WriteLn;
    Write('Presione Enter para continuar...');
    ReadLn;
end;

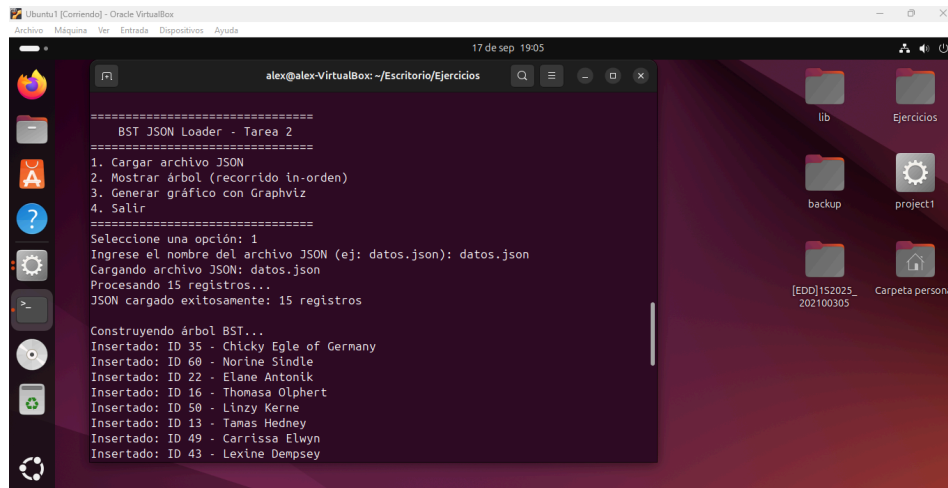
until opcion = 4;
end.
```



Gráfica generada:

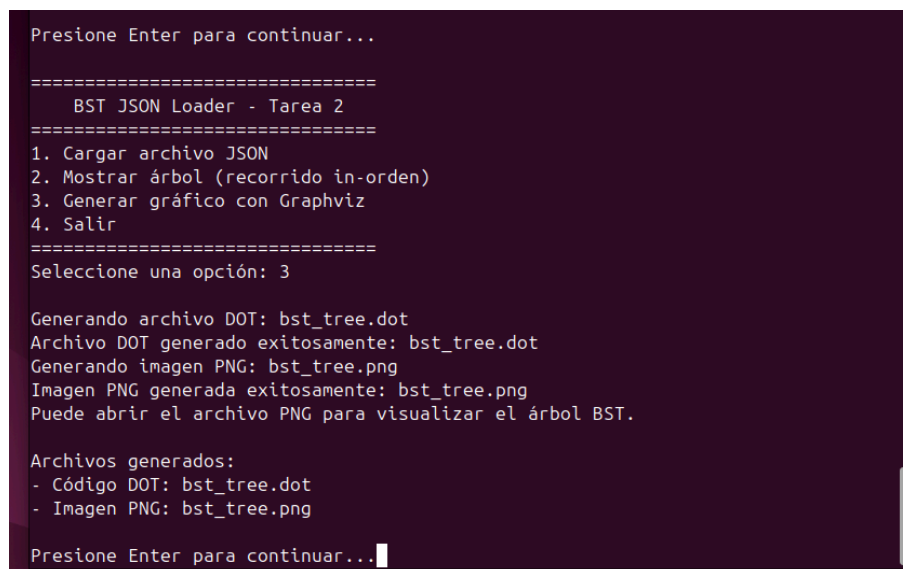


## Captura de la ejecución:



```
alex@alex-VirtualBox: ~/Escritorio/Ejercicios
=====
BST JSON Loader - Tarea 2
=====
1. Cargar archivo JSON
2. Mostrar árbol (recorrido in-orden)
3. Generar gráfico con Graphviz
4. Salir
=====
Seleccione una opción: 1
Ingrese el nombre del archivo JSON (ej: datos.json): datos.json
Cargando archivo JSON: datos.json
Procesando 15 registros...
JSON cargado exitosamente: 15 registros

Construyendo árbol BST...
Insertado: ID 35 - Chicky Egle of Germany
Insertado: ID 60 - Norine Sindle
Insertado: ID 22 - Elane Antonik
Insertado: ID 16 - Thomasa Olphert
Insertado: ID 50 - Linzy Kerne
Insertado: ID 13 - Tamas Hedney
Insertado: ID 49 - Carrissa Elwyn
Insertado: ID 43 - Lexine Dempsey
```



```
Presione Enter para continuar...

=====
BST JSON Loader - Tarea 2
=====
1. Cargar archivo JSON
2. Mostrar árbol (recorrido in-orden)
3. Generar gráfico con Graphviz
4. Salir
=====
Seleccione una opción: 3

Generando archivo DOT: bst_tree.dot
Archivo DOT generado exitosamente: bst_tree.dot
Generando imagen PNG: bst_tree.png
Imagen PNG generada exitosamente: bst_tree.png
Puede abrir el archivo PNG para visualizar el árbol BST.

Archivos generados:
- Código DOT: bst_tree.dot
- Imagen PNG: bst_tree.png

Presione Enter para continuar...
```