



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en ciencias de sistemas.

Manual de técnico

Nombre del estudiante

José Luis Espinoza Jolón - 202202182

Ing.(a). Kevin Lajpop

Guatemala, Abril 21 de 2024

Introducción

• Descripción del programa	3
• Lenguaje para utilizar	3
• Librería a utilizadas.....	3
▪ Jison :.....	3
Donde se descargar	3
▪ Graphviz-react:	3
Donde se descarga:.....	3
• Estructura del proyecto	4
❖ Backend-servicidor.....	5
Carpeta routes:	5
indexControle:	6
Carpeta controller :.....	6
lexico:	7
syntax:	7
Clases abstractas:	8
Clases arbol:	8
❖ My-app:	10

Descripción del programa

Aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador para la realización de un intérprete sencillo, con las funcionalidades principales para que sea funcional.

Lenguaje para utilizar

Para este tipo de proyecto utilizamos typescript.

Librería a utilizadas

- **Jison** : Jison es un generador de analizadores sintácticos (parser) diseñado para ayudar en la creación de intérpretes y compiladores. Es similar a Bison (Berkeley Yacc), pero está escrito en JavaScript y puede utilizarse tanto en el lado del servidor como en el navegador. Jison toma una gramática en forma de especificaciones de gramáticas libres de contexto y genera un parser en JavaScript que puede ser utilizado para analizar y procesar código fuente. Es una herramienta útil para proyectos que requieren análisis sintáctico de lenguajes de programación u otros formatos estructurados.

Donde se descargar: Link para descargar la librería

<https://gerhobbelt.github.io/jison/docs/>

- **Graphviz-react:**

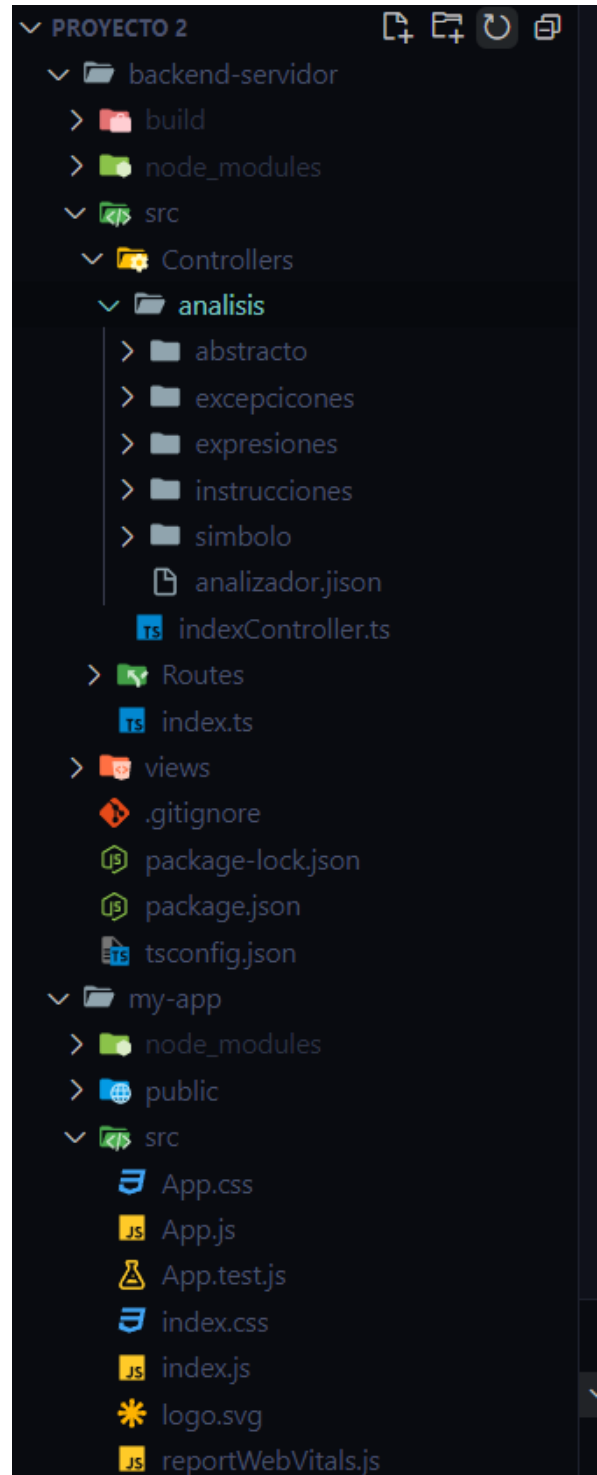
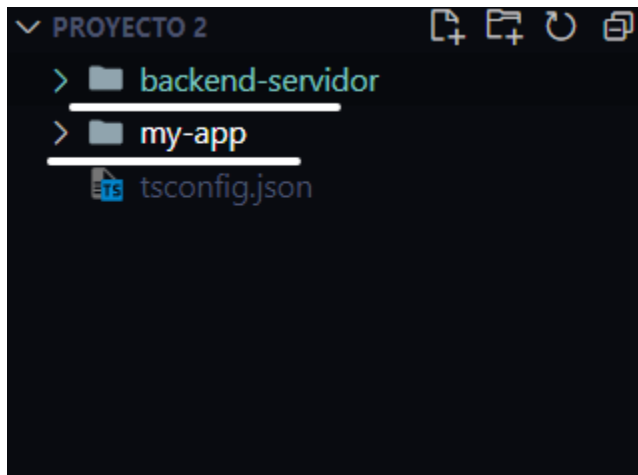
es una biblioteca de React que permite renderizar gráficos generados por Graphviz en aplicaciones web. Graphviz es una herramienta ampliamente utilizada para generar diagramas y visualizaciones de grafos a partir de descripciones de grafos en un lenguaje de texto llamado DOT.

Donde se descarga:

Link <https://www.npmjs.com/package/graphviz-react>

🚦 Estructura del proyecto

La manera que organice el proyecto es de la siguiente forma:



❖ Backend-servidor

En esta parte se realizó toda la parte lógica del programa, donde se trabajó la gramática del lenguaje y las funcionalidades

Carpeta routes:

En esta parte es donde trabajamos las rutas para manejarla desde el frontend

```
Gramatica - BNF(Backus-Naur.txt)  indexRouter.ts X
backend-servidor > src > Routes > indexRouter.ts > router > config

1  import { Router } from "express"
2  import { indexController } from "../Controllers/indexController"
3
4  class router {
5      public router:Router = Router();
6      constructor(){
7          this.config();
8      }
9
10     config(): void{
11         this.router.get('/',indexController.prueba);
12         this.router.post('/interpretar', indexController.interpretar)
13         this.router.get('/getAST', indexController.ast)
14     }
15 }
16
17
18 const indexRouter = new router();
19 export default indexRouter.router;
```

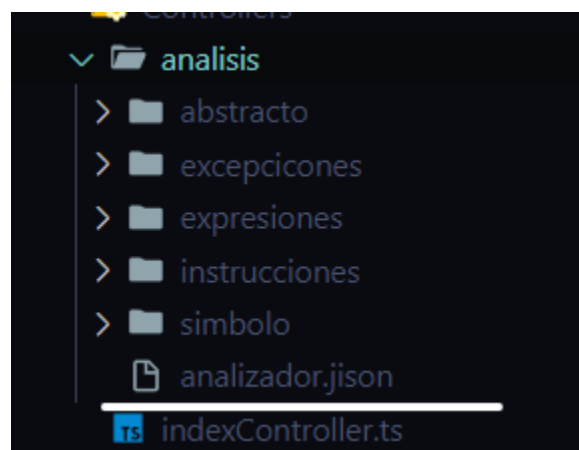
indexControle:

Donde realizamos la lectura de todo lo que viene de json

```
19 class controller {
20   public interpretar(req: Request, res: Response) {
21     errores_list = new Array<Errores>
22     try {
23       AstDot = ""
24       let parser = require('./analisis/analizador')
25       let ast = new Arbol(parser.parse(req.body.entrada))
26       let tabla = new tablaSimbolo()
27       tabla.setNombre("Ejemplo1")
28       ast.setTablaGlobal(tabla)
29       ast.setConsola("")
30
31       let execute = null;
32
33       //primer recorrdio
34       for(let i of ast.getInstrucciones()){
35         if(i instanceof Metodo || i instanceof Funcion){
36           i.id = i.id.toLocaleLowerCase()
37           ast.agregarFunciones(i)
38         }
39         if(i instanceof Declaracion){
40           i.interpretar(ast, tabla)
41           // manejo de errores
42         }
43         if (i instanceof Execute){
44           execute = i
45         }
46       }
47
48       for (let i of errores_list) {
49         ast.Print("----> "+i.getTipoError()+ ": " + i.getDescp() + " Fila: " + i.getFila()+ " Columna: " + i.getColumna()+"\n" )
50       }
51     }
52   }
53 }
```

Carpeta controller :

Donde se distribuyo las carpetas de cada parte del lenguaje y donde se realizo el json que es la gramática



lexico:

Se realizo todos los tokens del lenguaje

```
1 //comentario de una línea
2 [/](^)[^"]*(("[^"]*"|'[^']*'|\\(?!\\/))*[/])(.*) //multilinea
3
4 //palabras reservadas
5 "cout"      return 'IMPRIMIR'
6 "int"       return 'INT'
7 "double"    return 'DOUBLE'
8 "std::String" return 'STRING'
9 "true"      return 'TRUE'
10 "false"     return 'FALSE'
11 "char"      return 'CHAR'
12 "bool"      return 'BOOLEAN'
13 "pow"       return 'POW'
14 "tolower"   return 'TOLOWER'
15 "toupper"   return 'TOUPPER'
16 "round"     return 'ROUND'
17 "std::toString" return 'TOSTRING'
18 "if"        return 'IF'
19 "while"     return 'WHILE'
20 "break"     return 'BREAK'
21 "typeof"    return 'TYPEOF'
22 "length"    return 'LENGTH'
23 "else"      return 'ELSE'
24 "do"        return 'DO'
25 "for"       return 'FOR'
26 "continue" return 'CONTINUE'
27 "switch"    return 'SWITCH'
28 "case"      return 'CASE'
29 "default"   return 'DEFAULT'
30 "new"       return 'NEW'
31 "execute"   return 'EXECUTE'
32 "void"      return 'VOID'
33 "c_str"     return 'CSTR'
```

sintax:

Se realizo las producciones y la gramática

```
77 %left 'PUNTO'
78
79 // simbolo inicial
80 %start INICIO
81
82 %%
83
84 INICIO : INSTRUCCIONES EOF          {return $1;}
85 ;
86
87 INSTRUCCIONES : INSTRUCCIONES INSTRUCCION  {$1.push($2); $$=$1;}
88               | INSTRUCCION                {$$=[ $1 ];}
89 ;
90
91 INSTRUCCION : IMPRESION              {$$=$1;}
92             | DECLARACION PUNTOCOMA  {$$=$1;}
93             | ASIGNACION PUNTOCOMA    {$$=$1;}
94             | FUNIF                    {$$=$1;}
95             | FUNWHILE                  {$$=$1;}
96             | FUNBREAK                  {$$=$1;}
97             | FUNCONTINUE               {$$=$1;}
98             | FUNDOWHILE PUNTOCOMA      {$$=$1;}
99             | FUNFOR                     {$$=$1;}
100            | FUNSWITCH                   {$$=$1;}
101            | DECARREGLO PUNTOCOMA        {$$=$1;}
102            | MODVECTOR PUNTOCOMA         {$$=$1;}
103            | MODVECTOR2 PUNTOCOMA        {$$=$1;}
104            | DECARREGLO2DIMEN PUNTOCOMA  {$$=$1;}
105            | METODO {$$=$1;}
106            | FUNCION {$$=$1;}
107            | EXECUTEN PUNTOCOMA          {$$=$1;}
108
```

Clases abstractas:

Para poder realizar el proyecto realizamos las clases abstractas, en cada función para el programa

```
Gramatica - BNF(Backus-Naur-form) Instruccion.ts X
backend-servidor > src > Controllers > analisis > abstracto > Instruccion.ts > Instruccion > ArbolAST

1 import Arbol from "../simbolo/Arbol";
2 import tablaSimbolos from "../simbolo/tablaSimbolos";
3 import Tipo from "../simbolo/Tipo";
4
5
6 export abstract class Instruccion {
7     public tipoDato: Tipo
8     public linea: number
9     public col: number
10
11     constructor(tipo: Tipo, linea: number, col: number) {
12         this.tipoDato = tipo
13         this.linea = linea
14         this.col = col
15     }
16
17     abstract interpretar(arbol: Arbol, tabla: tablaSimbolos): any
18     abstract ArbolAST(anterior: string): string
19 }
20
21 )
```

Clases arbol:

Con esta clase se creo el árbol para poder manejar los datos

```
backend-servidor > src > Controllers > analisis > abstracto > Arbol.ts > Arbol > ArbolAST
1 import tablaSimbolo from "../tablaSimbolos";
2 import { Instruccion } from "../abstracto/Instruccion";
3 import Errores from "../excepciones/Errores";
4 import Metodo from "../instrucciones/Metodo";
5 import Funcion from "../instrucciones/Funciones";
6
7 export default class Arbol {
8     private instrucciones: Array<Instruccion>
9     private consola: string
10     private tablaGlobal: tablaSimbolo
11     private errores: Array<Errores>
12     private funciones: Array<Instruccion>
13
14     constructor(instrucciones: Array<Instruccion>) {
15         this.instrucciones = instrucciones
16         this.consola = ""
17         this.tablaGlobal = new tablaSimbolo()
18         this.errores = new Array<Errores>()
19         this.funciones = new Array<Instruccion>()
20     }
21
22     public Print(contenido: any) {
23         this.consola = `${this.consola}${contenido}`;
24     }
25
26     public getConsola(): string {
27         return this.consola
28     }
29
30     public setConsola(console: string): void {
31         this.consola = console
32     }
33 }
```


Clase tabla de simbol

Se creo la clase de símbolo para guardar los datos y poder manejarlo

```
backend-servidor > src > Controllers > analisis > simbolo > tablaSimbolos.ts > tablaSimbolo > setVan
1  import Simbolo from "../Simbolo";
2  import Tipom,{tipoDato} from "../Tipo";
3
4
5  export default class tablaSimbolo {
6      private tablaAnterior: tablaSimbolo | any
7      private tablaActual: Map<string, Simbolo>
8      private nombre: string
9
10     constructor(anterior?: tablaSimbolo) {
11         this.tablaAnterior = anterior
12         this.tablaActual = new Map<string, Simbolo>()
13         this.nombre = ""
14     }
15
16     public getAnterior(): tablaSimbolo {
17         return this.tablaAnterior
18     }
19
20     public setAnterior(anterior: tablaSimbolo): void {
21         this.tablaAnterior = anterior
22     }
23
24     public getTabla(): Map<String, Simbolo> {
25         return this.tablaActual;
26     }
27
28     public setTabla(tabla: Map<string, Simbolo>) {
29         this.tablaActual = tabla
30     }
31 }
```

Crear el Ast:

Se creo la siguiente clase stactic para poder manejarlo en todo nuestro proyecto

```
backend-servidor > src > Controllers > analisis > simbolo > AST.ts > Ast
1  export default class Ast {
2      private static instancia: Ast
3      private count: number
4
5      private constructor() {
6          this.count = 0
7      }
8
9      public static getInstancia(): Ast {
10         if (!Ast.instancia) {
11             Ast.instancia = new Ast()
12         }
13         return Ast.instancia
14     }
15
16     get() {
17         this.count++
18         return this.count
19     }
20 }
21 }
```

❖ My-app:

Clases app.js:

Se creo primero las funcionalidades de cada botón

```
import React, { useEffect, useState, useRef } from "react";
import './App.css';
import Editor from '@monaco-editor/react';
import { Graphviz } from 'graphviz-react';

function App() {
  const editorRef = useRef(null);
  const consolaRef = useRef(null);
  const [AST, obtenerAST] = useState("");

  const [archivos, setArchivos] = useState([]); // Estado para mantener los archivos
  const [archivoActual, setArchivoActual] = useState(null); // Estado para mantener

  function handleEditorDidMount(editor, id) {
    if (id === "editor") {
      editorRef.current = editor;
    } else if (id === "consola") {
      consolaRef.current = editor;
    }
  }

  function reporteAST(){
    fetch('http://localhost:4000/getAST', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
      },
    })
    .then(response => response.json())
    .then(data => {
      obtenerAST(data.AST);
      console.log(data.AST);
    });
  }
}
```

Se realizo la estructura del proyecto en html

```
return (
  <div className="App">
    <nav>
      <label className="logo">CompiScript</label>
      <ul>
        <li><a href="#" onClick={crearArchivoEnBlanco}>Crear archivo en blanco</a></li> The href attribute requires a valid value
        {archivos.map(archivo => (
          <li key={archivo.nombre}><a href="#" onClick={() => abrirArchivo(archivo.nombre)}>{archivo.nombre}</a></li> The href
        ))}
        <li><label className="nav-link" htmlFor="fileInput">Abrir Archivos</label>
        <input id="fileInput" type="file" accept=".sc" style={{ display: 'none' }} onChange={arbir_archivo_nuevo} /></li>
        <li><a href="#" onClick={guardarArchivo}>Guardar archivo</a></li> The href attribute requires a valid value to be access
        <li><a href="#" onClick={interpretar}>Ejecutar</a></li> The href attribute requires a valid value to be accessible. Prov
        <li><a href="#">Reportes</a></li> The href attribute requires a valid value to be accessible. Provide a valid, navigable
      </ul>
    </nav>

    <div className="editors-container">
      <h1>Entrada</h1>
      <div className="editor">
        <Editor height="90vh" width="100%" defaultLanguage="java" defaultValue="" theme="vs" onMount={(editor) => handleEditorDidMo
      </div>

      <h1>Consola</h1>
      <div className="editor-consola">
        <Editor height="90vh" width="100%" defaultLanguage="cpp" defaultValue="" theme="vs" options={{ readOnly: true }} onMount={{
      </div>
    </div>
  </div>
)
```

Clases app.css:

Se creo todo el estilo de la pagina

```
my-app > src > App.css nav label
1  * {
2    padding: 0;
3    margin: 0;
4    text-decoration: none;
5    list-style: none;
6  }
7
8  body {
9    background-color: #f8f8f8;
10   font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
11 }
12
13 .App {
14   display: flex;
15   flex-direction: column;
16   align-items: center;
17   padding: 20px; /* Agrega un espacio de relleno para separar los bordes de la página */
18 }
19
20 nav {
21   height: 80px;
22   width: 100%;
23   background: #094067;
24 }
25
26 label.logo {
27   font-size: 35px;
28   font-weight: bold;
29   color: white;
30   padding: 0 100px;
31   line-height: 80px;
```