

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 1

Proyecto 2  
Manual Técnico

César André Ramírez Dávila

202010816

Fecha: 04/11/2022

## Índice

<i>Introducción .....</i>	<b>3</b>
<i>Objetivos .....</i>	<b>3</b>
<i>Requerimientos .....</i>	<b>4</b>
<i>Creación del programa .....</i>	<b>5</b>
1. <b>Interfaz Gráfica .....</b>	<b>5</b>
2. <b>Carga de Archivo .....</b>	<b>5</b>
3. <b>Token de Errores .....</b>	<b>6</b>
4. <b>Analizador Lexico .....</b>	<b>6</b>
5. <b>Analizador Sintáctico.....</b>	<b>8</b>
6. <b>Clases abstractas .....</b>	<b>10</b>
7. <b>Gráfico del Árbol de sintaxis abstracta .....</b>	<b>12</b>
8. <b>Servidor .....</b>	<b>13</b>
9. <b>Frontend .....</b>	<b>15</b>

## Introducción

El presente documento describe los aspectos técnicos informáticos del programa. El documento familiariza al desarrollador que hace uso de node.js, con ayuda de las herramientas Jison y typescript con temas como: uso de Clases, Gramática, Analizador léxico, Analizador Sintáctico, cargas de archivos, servidor rest.

## Objetivos

- Instruir el uso adecuado del Sistema de información, para el acceso adecuado en el uso de este, mostrando los pasos de desarrollo del programa, así como la descripción de las funciones y métodos usados para la realizacion del programa.
- Comprender uso de las herramientas Jison y typescript.
- Obtener mayor conocimiento en node.js, angular y typescript.

El presente manual está enfocado para el manejo de un cliente-servidor.

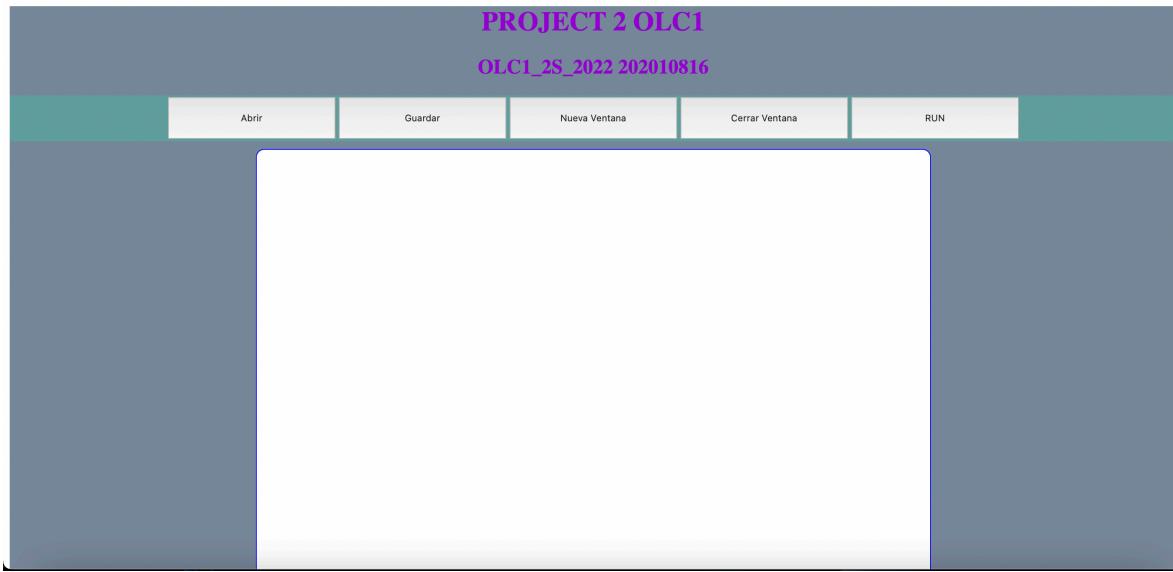
## Requerimientos

- La aplicación puede ser ejecutada en cualquier sistema operativo que tenga instalado node.js y jison en el sistema.
- IDE recomendado: Visual studio code
- Equipo Intel Pentium o superior
- Espacio en el disco duro, al menos 800 mb
- Memoria ram recomendada 4gb (por la conexión de frontend y backend)

## Creación del programa

### 1. Interfaz Gráfica

La primera parte del programa es tener una interfaz grafica donde se procederá a la ejecución de todo el programa.



Con la herramienta de Angular podemos hacer plantillas de html para el frontend

### 2. Carga de Archivo

Creamos la función que nos permite hacer la carga de un archivo con extension especifica .olc y poner su contenido en la caja de texto para su edición posteriormente.

```

CapturarArchivo(event:any, index:any){
  try {
    let a =event.target.files[0]
    let text="";
    if(a){
      let reader=new FileReader()
      reader.onload=ev=>{
        const resultado=ev.target?.result
        text=String(resultado)
        console.log(text.toString());
        this.entrada = text.toString();
        index.textarea.value = text.toString();
      }
      reader.readAsText(a)
    }
  } catch (error) {
    console.log(error);
  }
}

AbrirArchivo(archivo:HTMLInputElement){

  archivo.click();
}

```

### 3. Token de Errores

Obtenemos su tipo, lexema, descripcion, linea y columna

```

export class error {
  /**
   *
   * @param titulo Clasifica el tipo de error que se detecto, puede ser: lexico, sintactico o semantico
   * @param descripcion Detalle del error que se detecto
   * @param linea Linea en donde se encontro el error
   * @param columna Columna en donde se encontro el error
   */
  constructor(
    public titulo: string,
    public descripcion: string,
    public linea: number,
    public columna: number
  ) { }
}

```

### 4. Analizador Lexico

El siguiente paso para el programa es la creación del Analizador léxico con la ayuda de Jison

```
%{
    //Declaraciones y tambien las importaciones
const {Declaracion} = require('../instrucciones/Declaracion.ts');
const {Print} = require('../instrucciones/Print.ts');
const {PrintLn} = require('../instrucciones/PrintLn.ts');
const {Bloque} = require('../instrucciones/bloque.ts');
const {Asignacion} = require('../instrucciones/Asignacion.ts');
const {Casteo} = require('../instrucciones/Casteo.ts');
const {Incremento} = require('../instrucciones/Incremento.ts')
const {Decremento} = require('../instrucciones/Decremento.ts')
```

Declaraciones e importaciones que usaremos

```
%lex
%options case-insensitive

cadena [""][^"]*["]
caracter ([\'][^\n\']* [\'])
numero [0-9]+([.][0-9]+)?
bool "true"|"false"

%%

\s+          /* skip whitespace */
/*//".*           // comentario simple linea
[/* [*]*[*]+([/*][*]*[*]+)*/] // comentario multiple lineas

{cadena}      return 'cadena'
{numero}      return 'numero'
{bool}        return 'bool'
{caracter}    return 'char'
```

```
69  "String"   return 'pr_string'
70  "Int"      return 'pr_int'
71  "Boolean"  return 'pr_bool'
72  "Char"     return 'pr_char'
73  "Double"   return 'pr_double'
74  "Print"    return 'pr_print'
75  "Println"  return 'pr_println'
```

Terminales para el programa.

```

([a-zA-Z_]) [a-zA-Z0-9_ñ]*      return 'expreID';

<<EOF>>                      return 'EOF'

.
{
    let s= Singleton.getInstance()
    s.add_error(new error("Lexico","No se reconoce el caracter "+yytext,yylineno+1,yyloc.first_column+1));
}

```

Almacenando errores en una lista.

## 5. Analizador Sintáctico

```

/lex
%left '+'
%left '-'
%left '*'
%left '/'
%left '^'
%left '%'
%left ':'
%left '?'

```

Precedencia de operadores

```

%start INIT

%%

INIT : LISTAINSTRUCCIONES EOF {return $1;}
;
```

Iniciando Gramática

```

LISTAINSTRUCCIONES: LISTAINSTRUCCIONES INSTRUCCION { $1.push($2);   $$= $1;  }
|           |   INSTRUCCION {$$=$1}
;
```

## INSTRUCCION :

```
DECLARACIONES {$$=$1;}
| ASIGNACIONES {$$=$1;}
| CASTEO {$$=$1;}
| INCREMENTO {$$=$1;}
| DECREMENTO {$$=$1;}
| ENCAPSULAMIENTO {$$=$1;}
| VECTORES {$$=$1;}
```

Instrucciones de la gramática.

```
| error {let s= Singleton.getInstance()
        s.add_error(new error("Sintactico","No se esperaba el caracter "+yytext,yylineno+1,@1.first_column+1));}
;
```

Recuperando y almacenando errores sintácticos.

```
DECLARACIONES: TIPOS EXID '=' OPERACIONA ';' {$$= new Declaracion($2,$1,$4,@1.first_line,@1.first_column); }
| TIPOS EXID '=' 'cadena' ';' {$$= new Declaracion($2,$1,$4,@1.first_line,@1.first_column); }
| TIPOS EXID '=' 'char' ';' {$$= new Declaracion($2,$1,$4,@1.first_line,@1.first_column); }
| TIPOS EXID '=' 'bool' ';' {$$= new Declaracion($2,$1,$4,@1.first_line,@1.first_column); }
| TIPOS EXID ';' {$$= new Declaracion($2,$1,[],@1.first_line,@1.first_column); }
;
```

```
EXID: EXID ',' 'expreID' {$1.push($3); $$ = $1;}
| 'expreID' {$$ = [$1]}
;

ASIGNACIONES: EXID '=' OPERACIONA ';' {$$= new Asignacion($1, $3,@1.first_line,@1.first_column);}
| EXID '=' 'cadena' ';' {$$= new Asignacion($1, $3,@1.first_line,@1.first_column);}
| EXID '=' 'bool' ';' {$$= new Asignacion($1, $3,@1.first_line,@1.first_column);}
| EXID '=' 'char' ';' {$$= new Asignacion($1, $3,@1.first_line,@1.first_column);}
| EXID '=' EXID ';' {$$= new Asignacion($1, $3,@1.first_line,@1.first_column);}
;

OPERACIONA: OPERACIONA EXPRESIONES {$$=$1;}
| EXPRESIONES {$$=$1;}
;
```

```

ENCAPSULAMIENTO: '{' LISTAINSTRUCCIONES '}' { $$= new Bloque($2,@1.first_line,@1.first_column);}
| '{' 'cadena' ',' 'cadena' '}' { $$= new Bloque($4,@1.first_line,@1.first_column);}
;

VECTORES: DECLARARVECTOR {$$=$1;}
| DECLARARVECTOR2 {$$=$1;}
| ACCESOVECTOR {$$=$1;}
| MODIFICARVECTOR {$$=$1;}
;

DECLARARVECTOR: TIPOS '[' ']' EXID '=' 'pr_new' TIPOS '[' 'numero' ']' ';' { $$= new Vector($1,$4,$7,$9);}
| TIPOS '[' ']' '[' ']' EXID '=' 'pr_new' TIPOS '[' '(' TIPOS ')' 'cadena' ']' '[' 'numero' ']' ';' { $$= new Vector($1,$6,$9,$11);}
| TIPOS '[' ']' '[' ']' EXID '=' 'pr_new' TIPOS '[' 'numero' ']' '[' 'numero' ']' ';' { $$= new Vector($1,$6,$9,[' + $11+ ']);
;

DECLARARVECTOR2: TIPOS '[' ']' EXID '=' '{' LISTAVALORES '}' ';' { $$= new Vector2($1,$4,$7,null);}
| TIPOS '[' ']' '[' ']' EXID '=' '{' '{' LISTAVALORES '}' ',' '{' LISTAVALORES '}' '}' ';' { $$= new Vector2($1,$6,$10,$14);}
;

ACCESOVECTOR: TIPOS EXID '=' 'expreID' '[' 'numero' ']' ';' { $$= new AccesoVector($1,$2,$4,$6,null);}
| TIPOS EXID '=' 'expreID' '[' 'numero' ']' '[' 'numero' ']' ';' { $$= new Vector($1,$2,$4,$6,$9);}
;

MODIFICARVECTOR: EXID '[' 'numero' ']' '=' 'cadena' ';' { $$= new ModificarVector($1, $3, $6, null);}
| EXID '[' 'numero' ']' '=' 'cadena' '+' 'expreID' '[' 'numero' ']' ';' { $$= new ModificarVector($1,$3,$6 + $8, $10);}
;

TIPOS: 'pr_int' {$$=$1;}
| 'pr_char' {$$=$1;}
| 'pr_string' {$$=$1;}
| 'pr_bool' {$$=$1;}
| 'pr_double' {$$=$1;}
;

```

Continuando con todas las instrucciones sucesivamente

## 6. Clases abstractas

```

1 import nodo from "../Abstract/nodo";
2 import { Env } from "../symbols/env";
3
4 export abstract class Instruccion{
5     //atributos
6     constructor(public line: number, public column: number) {
7         this.line = line;
8         this.column = column;
9     }
10
11     public abstract ejecutar():any;
12     public abstract getNode():nodo;
13 }
14

```

Clase para las instrucciones

```

1 import { Instruccion } from "../abstractas/instruccion";
2 import { Env } from "../symbols/env";
3 import nodo from "./Abstract/nodo";
4 export class Asignacion extends Instruccion {
5
6     constructor(
7         public nombre: string[],
8         public contenido: string,
9         linea: number, columna:number) {
10        super(linea,columna);
11    }
12
13    public ejecutar():any {
14        console.log("Encontre una asignacion, nombre:"+this.nombre +", contenido: " + this.contenido + " lo encontre en la linea "+this.linea);
15    }
16
17    public getNode() {
18        var nodoAsig = new nodo("<ASIGNACION>");
19        this.nombre.forEach(id => {
20            nodoAsig.agregarHijo(id);
21        });
22        nodoAsig.agregarHijo(this.contenido);
23        return nodoAsig;
24    }
25
26 }

```

Creando las instrucciones que recibe la clase del JSON.

```

1 import { Instruccion } from "../abstractas/instruccion";
2 import { Env } from "../symbols/env";
3 import nodo from "./Abstract/nodo";
4
5 export class Incremento extends Instruccion {
6
7     constructor(
8         public nombre: string,
9         public contenido: string,
10        linea: number, columna:number) {
11        super(linea,columna);
12    }
13
14    public ejecutar():any {
15        //console.log("Encontre una asignacion, nombre:"+this.nombre+" lo encontre en la linea "+this.linea);
16        console.log("Encontré un incremento, nombre:" +this.nombre);
17    }
18
19    public getNode() {
20        var nodoInc = new nodo("<INCREMENTO>");
21        nodoInc.agregarHijo(this.nombre);
22        nodoInc.agregarHijo(this.contenido);
23        return nodoInc;
24    }
25
26 }

```

Sucesivamente con cada instrucción que nuestro programa acepta.

En INSTRUCCIÓN tenemos los siguientes no terminales

- Declaraciones
- Asignaciones
- Casteo
- Incremento

- Decremento
- Ciclo IF
- Ciclo Switch
- Ciclo For
- Ciclo While
- Ciclo Do While
- Retorno
- Metodo
- Funciones
- Ejecutar
- Print
- Println

## 7. Gráfico del Árbol de sintaxis abstracta

```

var grafo = "";
grafo = getDot(instrucciones);
//console.log(grafo);
var dot = "";
var c = 0;

function getDot(raiz: nodo) {
    dot = "";
    dot += "digraph grph {\n";
    dot += 'nodo0[label=' + raiz.getValor().replace("'", "\\\\'") + "];\n";
    c = 1;
    recorrerAST("nodo0", raiz);
    dot += "}";
    writeFile("prueba.dot", dot)
    exec('dot -Tpng prueba.dot -o servidor.png ')
    //exec('dot -Tsvg prueba.dot -o servidor.svg')
    return dot;
}

function recorrerAST(padre: String, nPadre: nodo) {
    for (let hijo of nPadre.getHijos()) {
        var nombreHijo = "nodo" + c;
        var primerquite = hijo.getValor().toString().replace("\'", " ");
        dot += nombreHijo + "[label=" + primerquite.replace("\'", " ") + "]\n";
        dot += padre + "->" + nombreHijo + ";\n";
        c++;
        recorrerAST(nombreHijo, hijo);
    }
}

function writeFile(nameFile: string, data: string) {
    fs.writeFile(nameFile, data, () => {
        console.log('>> The file ' + nameFile + ' has been saved!');
    })
}

```

## 8. Servidor

Se usó un servidor en typescript que conecta con angular.

```
1 import express, { Application } from "express";
2 import morgan from "morgan";
3 import cors from "cors";
4
5 import apiRoutes from "./routes/apiRouters";
6
7 class Server {
8   public app: Application;
9
10  constructor() {
11    this.app = express();
12    this.config();
13    this.routes();
14  }
15
16  config(): void {
17    this.app.set("port", process.env.PORT || 8080);
18    this.app.use(morgan("dev"));
19    this.app.use(cors());
20    this.app.use(express.json());
21    this.app.use(express.urlencoded({ extended: false }));
22  }
23
24  routes(): void {
25    this.app.use("/usuario", apiRoutes);
26    this.app.use("/prueba", apiRoutes);
27  }
28
29
30
31  start(): void {
32    this.app.listen(this.app.get("port"), () => {
33      console.log("Server on port ", this.app.get("port"));
34    });
35  }
36
37
38 export const server = new Server();
```

Iniciando el servidor con las rutas para la conexión

```

1 import { Router } from "express";
2
3 //import { apiController } from "../controllers/apiController";
4 import { apiController } from "../controllers/apiController"
5
6 class ApiRoutes {
7   public router: Router = Router();
8
9   constructor() {
10     this.config();
11   }
12
13   config(): void {
14     this.router.get("/", apiController.funcion1);
15     this.router.post("/", apiController.funcion2);
16     this.router.post("/errores", apiController.funcion3);
17     this.router.get("/errores", apiController.funcion4);
18     //this.router.post("/", apiController.funcion3);
19   }
20 }
21
22 const apiRoutes = new ApiRoutes();
23 export default apiRoutes.router;

```

Las rutas que tendremos

```

178   public async funcion4(req: Request, res: Response) {
179     const s= Singleton.getInstance();
180     const fs = require("fs");
181     writeFile("errores.html", s.get_error())
182     let html = ` <div>
183       ${s.get_error()}
184     </div>
185     `;
186
187     function writeFile(nameFile: string, data: string) {
188       fs.writeFile(nameFile, data, () => {
189         console.log('>> The file ' + nameFile + ' has been saved!');
190       })
191     }
192     //res.send(html);
193     res.send({html:html});
194     s.clean();
195   }
196
197 }

```

Ejemplo de un método get que da respuesta para el frontend.

## 9. Frontend

El frontend está desarrollado en angular CLI versión 14.2.5

```
1 <div class="header">
2   <h1 class="titulo">PROJECT 2 OLC1</h1>
3   <h2 class="titulo"><b>OLC1_2S_2022 202010816</b></h2>
4   <nav class="navegacion">
5     <input style="display:none;" accept=".olc" type="file" (change)="CapturarArchivo($event, sendForm.value)" #archivo>
6     <button class="boton" mat-raised-button color="primary" (click)=AbrirArchivo(archivo)>Abrir</button>
7     <button class="boton" mat-raised-button color="accent" (click)=GuardarArchivo(sendForm.value)>Guardar</button>
8     <button class="boton" mat-raised-button color="primary">Nueva Ventana</button>
9     <button class="boton" mat-raised-button color="accent">Cerrar Ventana</button>
10    <button class="boton" mat-raised-button color="primary" (click)=setdata(sendForm.value)">RUN</button>
11    <form #sendForm="ngForm">
12      <textarea name="textarea" id="textarea" class="editor-container" style="width:800px;height:600px; border:1px solid blue">
13    </form>
14  </nav>
15 </div>
16 <div class="body-app">
17   <div id="errors"></div>
18 </div>
19 <div class="footer-app">
20   <button class="boton-foot" mat-raised-button color="primary">Reporte AST</button>
21   <button class="boton-foot" mat-raised-button color="accent">Ver Tabla De Simbolos</button>
22   <button class="boton-foot" mat-raised-button color="primary" (click)=VerErrores1()>Ver Tabla De Errores</button>
23 </div>
24 <router-outlet></router-outlet>
```

Salida en html

```
1 import { Component, OnInit } from '@angular/core';
2 import { UserService } from 'src/app/services/user.service';
3 import { Router } from '@angular/router';
4 import { ModalModule } from 'ngx-bootstrap/modal';
5
6 export interface Errores{
7   line:number,
8   column:number,
9   type:number,
10  message:string
11 }
12
13 interface Ventana{
14   nombre:string;
15   code:string;
16 }
17
18 @Component({
19   selector: 'app-dashboard',
20   templateUrl: './dashboard.component.html',
21   styleUrls: ['./dashboard.component.css']
22 })
23 export class DashboardComponent implements OnInit {
24   public archivos: any = []
25   entrada: string;
26   salidaeeee: string = "";
27   ImagePath: string = "";
28   ASTstring:string = "";
29
30   constructor(private service: UserService, private _router:Router) { this.entrada = "de unaaaa" }
31
32   ngOnInit(): void {
33 }
```

Components.ts

```
setdata(values:any):void{
    console.log("values", values.txtarea);
    var json={
        "peticion": values.textarea
    }
    this.service.setdata(json).subscribe(
        (res)=>{
            console.log(res);
            alert("Archivo Analizado con exito")
        }, (err)=>{
            console.log(err);
        }
    )
}
```

Con este método podemos analizar el contenido del editor de texto en el backend.

```
setdata(json:any){
    return this.http.post(` ${this.URL}/prueba`,json);
}
```

Mandando petición al servidor.

Así sucesivamente con los demás métodos que tiene el servidor.

## LINK REPOSITORIO

<https://github.com/AndreRD1026/OLC1-202010816>