Universidad de San Carlos de Guatemala
Facultad de ingeniería
Ingeniería en Ciencias y Sistemas
ORGANIZACIÓN DE
LENGUAJES Y
COMPILADORES

MANUAL TECNICO

Alberto Josue Hernández Armas 201903553

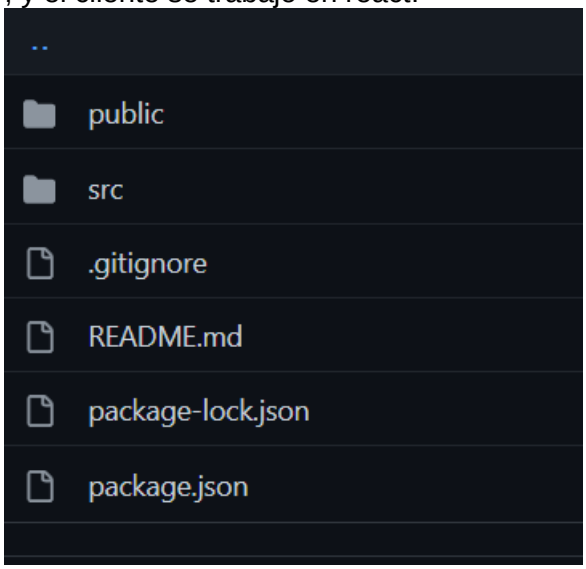
Guatemala 4 de NOVIEMBRE
de2022

INTRODUCCION

En el presente manual técnico podremos observar cómo es que está diseñado el código y explicando que forman la funcionalidad de una manera general, teniendo como objetivo principal que sea más entendible. Determinando los métodos utilizados y explicando algunas palabras claves refiriéndonos a sus propiedades demostrando cuál es su función dentro de los bloques de código que se nos presentan a continuación.

MANUAL TECNICO

Este proyecto se trabajo como una API, cliente servidor, el servidor esta hecho en el framework node js
, y el cliente se trabajo en react.



Del salo del servidor se trabajo todo lo que es el manejo de la gramática, se tienen loarchivos tipo jison, así como
Los objetos que dan funcionalidad a esta gramática, los objetos sobre los que depende el ast de cada gramática.

```
//palabras reservadas

"var"    return 'pr_var'
"let"    return 'pr_let'
"const" return 'pr_const'

"int" return 'pr_numero'
"double" return 'pr_double'
"char" return 'pr_char'
"string" return 'pr_string'
"boolean" return 'pr_bool'
"print" return 'pr_print'
"println" return 'pr_println'
"return" return 'pr_return'
"while" return 'pr_while'
"switch" return 'pr_switch'
"for" return 'pr_for'
"do" return 'pr_do'
"if" return 'pr_if'
"else" return 'pr_else'
"elif" return 'pr_elif'
"break" return 'pr_break'
"void" return 'pr_void'
"call" return 'pr_call'
"typeof" return 'pr_typeof'
"case" return 'pr_case'
"default" return 'pr_default'
"until" return 'pr_until'
"continue" return 'pr_continue'
"tolower" return 'pr_TL'
"toupper" return 'pr_TU'
"round" return 'pr_round'
"length" return 'pr_len'
"tostring" return 'pr_TS'
```

Análisis lexico de la gramática.

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


INSTRUCCIONES :   INSTRUCCIONES INSTRUCCION {$1.push($2); $$=$1;   console.log("s ")}
              |   INSTRUCCION               {$$ = [$1]; console.log("s ") }
              ;


INSTRUCCION : DECLARACION   {$$=$1; console.log("reconocio declaracion ") }
            | IMPRIMIR      {$$=$1; console.log("reconocio PRINT ") }
            | IMPRIMIRLN    {$$=$1; console.log("reconocio PRINTLN ") }
            | ASIGNACION    {$$=$1; console.log("reconocio asignacion ") }
            | METODO        {  console.log("reconocio metodo")}
            | FUNCION       { console.log("reconocio funcion") }
            | METODOsp        {  console.log("reconocio metodo sin parametros")}
            | FUNCIONsp       { console.log("reconocio funcion sin parametros") }
            | CONDICIONIF   { console.log("reconocio condicion if") }
            | CICLO         {console.log("reconocio  ciclo")}
            | RETURN   {console.log("reconocio  RETURN")}
            | CALL ';'     {console.log("reconocio  LLAMADA")}
            | SWITCH    {console.log("reconocio sentencia SWITCH")}
            | BREAK     {console.log("reconocio sentencia BREAK")}
            | CONTINUE     {console.log("reconocio sentencia CONTINUE")}
            | AUMENTO ';'   {$$=$1;console.log("reconocio sentencia AUMENTO")}
            | INSTANCIA ';'   {$$=$1;console.log("reconocio sentencia INSTANCIA")}
            |DECLARACION_VECTORES {$$=$1;console.log("reconocio sentencia DECLARACION VECTOR")}
            | error    ';'  { console.log("Error sintactico en la linea"+(yylineno+1));
                                consola.cons+="\nERROR: Error sintactico en la linea"+(yylineno+1)+"\n" }
;
//INSTRUCCIONES CICLOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOS
```

```
194  ;
195  //INSTRUCCIONES CICLOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOS
196  CICLO: 'pr_for'  '(' DECLARACION ETS ';'  ETS ')' '{' INSTRUCCIONES '}'  {}
197      | 'pr_for' '('  ASIGNACION ETS ';'  ETS   ')'  '{' INSTRUCCIONES '}'  {}
198      |'pr_for'  '(' DECLARACION ETS ';'  ASIGNACION ')' '{' INSTRUCCIONES '}'  {}
199      | 'pr_for' '('  ASIGNACION ETS ';'  ASIGNACION   ')'  '{' INSTRUCCIONES '}'  {}
200      | 'pr_while' '(' ETS ')' '{' INSTRUCCIONES '}'  {}
201      | 'pr_do' '{' INSTRUCCIONES '}' 'pr_while' '(' ETS ')' ';' {}
202      | 'pr_do' '{' INSTRUCCIONES '}' 'pr_until' '(' ETS ')' ';' {}
203      ;
204
205  SWITCH : 'pr_switch' '(' 'id' ')' '{' OPCIONES 'pr_default' ':' '{' INSTRUCCIONES '}' '}' {}
206  ;
207  OPCIONES: OPCIONES OPCION  {}
208              | OPCION  {}
209  ;
210  OPCION : 'pr_case'  ETS  ':' '{' INSTRUCCIONES '}' {}
211  ;
212
213  BREAK: 'pr_break' ';' {}
214  ;
215  CONTINUE: 'pr_continue' ';' {}
216  ;
217
218
219
```

```
IDS : IDS ',' 'id' {$1.push($3); $$=$1;}
    | 'id'{$$ = [$1]}
    ;
COMPARACIONES: '!' '(' COMPARACIONES ')' {$$= new Relacional(null,$2,RelacionalOption.NEGACION, @1.first_line, @1.first_column);}
            |  COMPARACIONES '&&' COMP {$$= new Relacional($1,$3,RelacionalOption.AND, @1.first_line, @1.first_column);}
            |  COMPARACIONES '||' COMP {$$= new Relacional($1,$3,RelacionalOption.OR, @1.first_line, @1.first_column);}
            |   COMP  {$$=$1;}
;
COMP:  E '<' E {$$= new Relacional($1,$3,RelacionalOption.MENOR, @1.first_line, @1.first_column);}
    |  E '>''=' E  {$$= new Relacional($1,$3,RelacionalOption.MAYORIGUAL, @1.first_line, @1.first_column);}
    |  E '<''=' E  {$$= new Relacional($1,$3,RelacionalOption.MENORIGUAL, @1.first_line, @1.first_column);}
    |  E '>' E     {$$= new Relacional($1,$3,RelacionalOption.MAYOR, @1.first_line, @1.first_column);}
    |  E '!''=' E   {$$= new Relacional($1,$3,RelacionalOption.NOIGUAL, @1.first_line, @1.first_column);}
    |  E '=''=' E   {$$= new Relacional($1,$3,RelacionalOption.IGUAL, @1.first_line, @1.first_column);}
;



E: E '+' Term {$$= new Arithmetic($1,$3,AritmeticOption.MAS, @1.first_line, @1.first_column);}
|E '-' Term {$$= new Arithmetic($1,$3,AritmeticOption.MENOS, @1.first_line, @1.first_column);}
|'-' Term {{$$= new Arithmetic(new Literal("0",Type.NUMBER , @1.first_line, @1.first_column),$2,AritmeticOption.MENOS, @1.first_line, @1.first_column);} }
| AUMENTO {$$=$1;}
//| E '-' '-' {}
//|CALL {}
|Term  {$$=$1;}
;

Term: Term '*' Factor {$$= new Arithmetic($1,$3,AritmeticOption.MULTIPLICACION, @1.first_line, @1.first_column);}
|Term '/' Factor {$$= new Arithmetic($1,$3,AritmeticOption.DIVISION, @1.first_line, @1.first_column);}
| Term '%' Factor  {$$= new Arithmetic($1,$3,AritmeticOption.MODULO, @1.first_line, @1.first_column);}
|  Term '^' '(' E ')' {$$= new Arithmetic($1,$4,AritmeticOption.POTENCIA, @1.first_line, @1.first_column);}
```

Análisis sintáctico de la gramática:
En este caso no se utilizaron las precedencias predeterminadas en la librería, se utilizo LR1 para las precedencias, esto facilito en gran manera la visualización del AST, se recomienda utilizarlo en vez de precedencias predeterminadas.

Se hicieron dos analizadores sintácticos, la gramática es la misma, con la única diferencia de que uno es para la graficacion del ast y la otra es para el análisis semántico.

El método de graficacion se trabajo por medio de una lista enlazada, la cual contiene un algoritmo que detecta el tipo de dato dentro de la misma para conectar los nodos de manera recursiva siguiendo la lógica de conjuntos de la gramática.

```
v( x,  c)// x y c son nodos
{

    var exemple = "";
    var listaexemple = new listaenlazada();
    while (c!= null)
    {
        if (x.value.constructor.name == exemple.constructor.name && c.value.constructor.name==listaexemple.constructor.name)
        {

            var aptd = c.value.First; //
            while (aptd != null)
            {
                if (aptd.value.constructor.name == "String") {


                    if (!this.rep.includes("\n" + x.index + "->" + aptd.index)) {
                        this.conexiones += "\n" + x.index + "->" + aptd.index;
                        this.rep.push("\n" + x.index + "->" + aptd.index);
                    }
                }
                else {


                    this.v(aptd.Prev, aptd);
                }
                aptd=aptd.Next;
            }

        }

        x = x.Next;
        c = c.Next;

    }
```

También se tiene la parte funcional del servidor, donde se instancian los endpoints y las funciones de cada uno.

```
app.post('/setIncremental', function(req,res){
    var texto = req.body.texto
    console.log(texto)
    var salida = arbol_graf.parse(texto).toString();
    fs.writeFile('./salida_graphviz.txt', salida, err => {
        if (err) {
          console.error(err);
        }
        // file written successfully
      });
    res.json({status: "ok",
    incremental: incremental,
    salida : salida
    })

})
app.post('/gramaticarlos', function(req,res){
    var env_padre = new Environment(null);
    var texto = req.body.texto
    console.log(texto)
    const ast = parser.parse(texto);
        for (const elemento  of ast) {
            try {

                //preguntar si ese elemtno es de clase metodo o funciones

                    elemento.executar(env_padre)

            } catch (error) {
                //console.log(error);

            }
        }
        const obj = Object.fromEntries(env_padre.tablaSimbolos);
```

Ahora veremos la parte del cliente:
Fue realizada en react, lo que facilito la instancia de objetos y funciones que interactúen con la interfaz grafica, además de facilitar la conexión con el servidor

Esta es la parte principal funcional de react:

```
*/
import React, { Component } from 'react'
import './App.css'
import PostForm from './components/PostForm'
import PostForm2 from './components/PostForm2'

class App extends Component {
        render() {
                return (



                        <div className="App">
                                <PostForm />
                                {/* <PostList /> */}
        <PostForm2 />
                                {/* <PostList /> */}
                        </div>
                )
        }
}

export default App
```

```
import axios from 'axios'
//import { response } from 'express'

 class PostForm extends Component {
    constructor(props) {
      super(props)

      this.state = {
         texto:''
      }
    }
    changeHandler = e => {
        this.setState({[e.target.name]: e.target.value})
    }
    submitHandler = e  => {
        e.preventDefault()
        console.log(this.state)
        axios.post('http://localhost:5000/setIncremental',this.state).then
        (response=>{console.log((response.data.salida).toString());
        alert((response.data.salida).toString())
        }).catch(error => {
            console.log(error)
        })
    }
   render() {
    const {texto} = this.state
    return (
      <div>
        <form onSubmit={this.submitHandler}>
        <div>
        <label>EDITOR DE TEXTO:</label>
            </div>
            <div>
            <textarea type="text"  name="texto" rows="20" cols="80" value={texto} onChange={this.changeHandler}/>
            </div>
            <div>
                <button type='submit'>GRAFICAR AST!!!!</button>
            </div>
```

```
import axios from 'axios'
//import { response } from 'express'

 class PostForm2 extends Component {
    constructor(props) {
      super(props)

      this.state = {
         texto:''

      }
      this.mjs = "";
      this.response = "";
    }
    changeHandler = e => {
        this.setState({[e.target.name]: e.target.value})
        this.msj = this.response;
    }

    submitHandler = e  => {
        e.preventDefault()
        console.log(this.state)
        axios.post('http://localhost:5000/gramaticarlos',this.state).then
        (response=>{

        console.log(response.data.salida);
        //alert((response.data.salida).toString())




        this.response  = JSON.stringify(response.data.salida, null, 4);
        alert(JSON.stringify(response.data.consola, null, 4) )



        }).catch(error => {
            console.log(error)
```

Y así concluye la funcionalidad del programa.