UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

MANUAL TÉCNICO
Compscript

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1
PROYECTO 2

POR:

WALTER GUSTAVO COTI XALIN
201700522

LIBRERIAS
BACKEND

- Nodemon
- Jison
- Cors
- Copy
- Morgan
- Typescript
- Express

FRONTEND

- Material-ui core
- Material-ui icons

FRAMEWORK
React

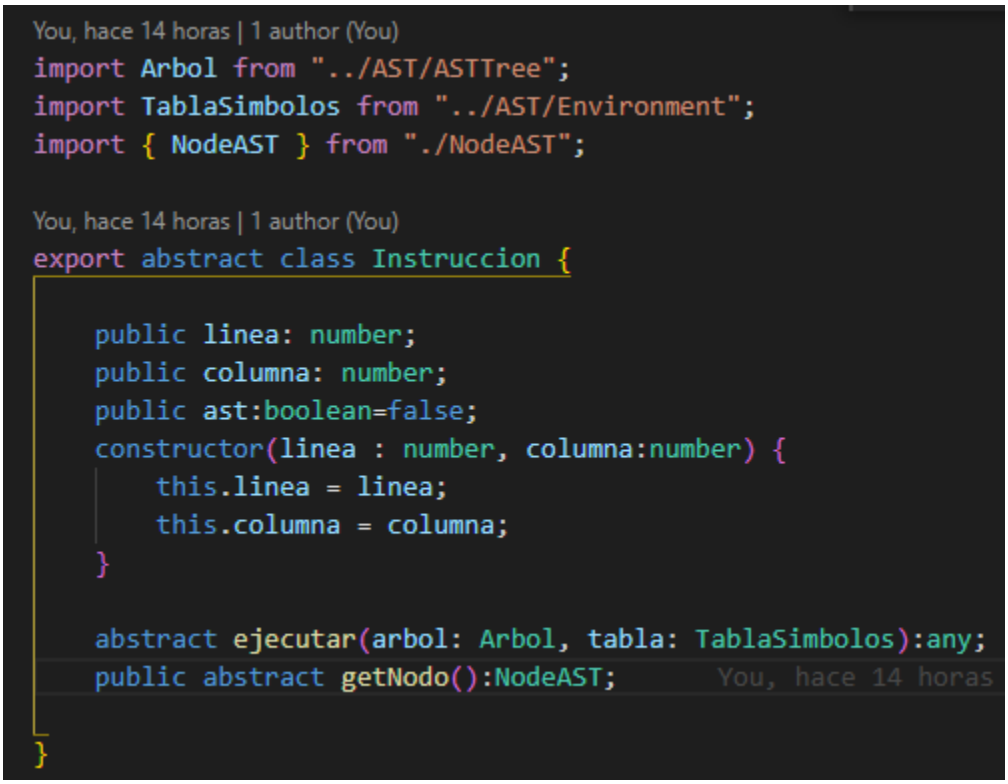
SCRIPT UTILIZADOS
Inicializar Backend:

```
"jison": "jison ./src/Analizador/analizador.jison" // COMPILE PROYECT
"copy": "copy analizador.js build/src/Analizador"  //COMPILE PARSER
"dev": "nodemon ./build/index.js"            //COPY PARSER IN BUILD
```

Inicializar Front-end:
Npm run build
Npm run start

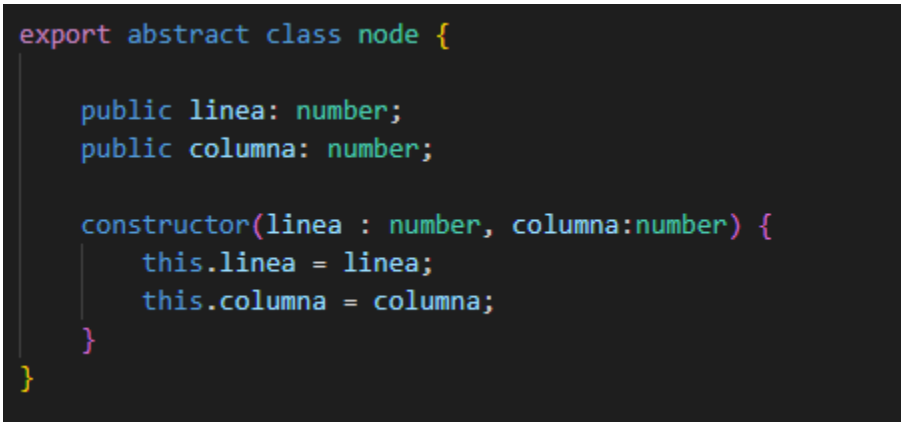
Objetos Abstractos

INSTRUCIONES
Objeto utilizado para las funciones e instrucciones.



NODO

Nodo auxiliar, utilizado en la lectura de el editor web.



NODOAST

Nodo auxiliar del AST, sus parametros

```
constructor(valor:string) {
    this.valor = valor;
}

public setHijos(hijos:Array<NodeAST>){
    this.hijos = hijos;
}

public agregarHijo(cad?:string, hijos?:Array<NodeAST>, hijo?:NodeAST){
    if (cad) {
        this.hijos.push(new NodeAST(cad));
    }
    else if (hijos) {
        for(let hijo of hijos)
        {
            this.hijos.push(hijo);
        }
    }else if(hijo){
        this.hijos.push(hijo);
    }
}
public agregarPrimerHijo(cad?:string, hijo?:NodeAST)
{
    if (cad) {
        this.hijos.unshift(new NodeAST(cad));
    }else if (hijo) {
        this.hijos.unshift(hijo);
    }
}
```

El programa inicia con la interpretación léxica y sintáctica de la entrada por medio del parser producido por jison, esta interpretación genera una lista de objetos de tipo instrucción y una lista de errores sintácticos-lexicos (Si se encuentran), en caso los errores sean irrecuperables se reportan y el programa termina, en caso sean recuperables se reportan y en caso no hayan sigue la ejecución.
En este momento pasa por una interpretación semántica en la cual la primera pasada se encarga de la declaración de todos los métodos y variables globales, posteriormente una segunda pasada ejecutara el Run, en esta ejecución se ven errores semánticos los cuales se van agregando a la lista de errores para ser reportados.
Una vez se haya interpretado y se hayan realizado las acciones se procede a graficar el AST y la tabla de símbolos ´por medio de los métodos de cada clase.

```
for(let m of ast.getInstrucciones()){
    if(m instanceof Excepcion){ // ERRORES SINTACTICOS
        ast.errores.push(m);
        ast.updateConsola((<Excepcion>m).toString());
        continue;
    }

    //Comprobara que sean declaraciones para ignorarlas y exec para ejecutar
    if((m instanceof Declaracion) || (m instanceof Asignacion) || (m instanceof DeclaracionFunciones) ||
    (m instanceof DeclaracionLista) || (m instanceof DeclaracionLista) || (m instanceof DeclaracionMetodos)
    || (m instanceof DeclaracionVector) || (m instanceof Exec) || (m instanceof Nativas)){
        if((m instanceof Exec)){
            if(comprobacionExec){
                comprobacionExec = false;
                let result = m.interpretar(ast, tabla);
                if(result instanceof Excepcion){ // ERRORES SINTACTICOS
                    ast.errores.push(result);
                    ast.updateConsola((<Excepcion>result).toString());
                }
            }else{
                let error = new Excepcion("Semantico", "Instruccion Exec llamada en mas de una ocasion", m.linea, m.columna);
                ast.errores.push(error);
                ast.updateConsola(error.toString());
            }
        }
    }else{
        let error = new Excepcion("Sintactico", "Instruccion fuera de un entorno de ejecucion", m.linea, m.columna);
        ast.errores.push(error);
        ast.updateConsola(error.toString());
    }
```

Cuando este se encuentra interpretado y se hayan realizado las acciones se procede a graficar el AST y la tabla de símbolos ´por medio de los métodos de cada clase.

```
var inicial = new NodoAST("RAIZ");
var instrucciones = new NodoAST("INSTRUCCIONES");
for(let m of ast.getInstrucciones()){
    try{
        instrucciones.agregarHijoNodo(m.getNodo());
    }catch(ex){
        instrucciones.agregarHijo("Error");
    }
}
inicial.agregarHijoNodo(instrucciones);
raiz = inicial;
try{
    var dotsito = graphAST();
}catch(Exception){
    var dotsito = "digraph {\n n[label=\"Errores detectados\"];}"
}

var simbolosdot = recorrerTablaSimbolos(ast.listaTablas);

var tablaerrores = TablaErrores(ast.errores);

res.json({valor:ast.getConsola(), dot:dotsito, tabladot:simbolosdot, doterrores:tablaerrores});
```