

Manual tencico

```
//options case-insensitive

bool "true" | "false"

%%
\s+          /* skip whitespace */
"--".*       // comment a line
[/][*][^]*[*]+([/*][^]*[*]+)*[/] // comment multiple lines

(\d{4})"-"(\d{1,2})"-"(\d{1,2})    {data.tokens.push({Tipo: 'TK_VALUE_DATE', Valor: yytext}); return 'TK_VALUE_DATE';}
(\\"(\\".|\\"\\")*\")              {yytext = yytext.substr(1, yytext.length-2); return 'TK_VALUE_VARCHAR';}
{bool}                            {data.tokens.push({Tipo: 'TK_VALUE_BOOLEAN', Valor: yytext}); return 'TK_VALUE_BOOLEAN';}
([0-9]+)"."([0-9]+)              {data.tokens.push({Tipo: 'TK_VALUE_DOUBLE', Valor: yytext}); return 'TK_VALUE_DOUBLE';}
[0-9]+                            {data.tokens.push({Tipo: 'TK_VALUE_INT', Valor: yytext}); return 'TK_VALUE_INT';}

"*"                               {data.tokens.push({Tipo: 'ASTERISCO', Valor: yytext});return '*';}
"/"                               {data.tokens.push({Tipo: 'SLASH', Valor: yytext});return '/';}
","                               {data.tokens.push({Tipo: 'PUNTO_COMA', Valor: yytext});return ',';}
"."                               {data.tokens.push({Tipo: 'DOS_PUNTOS', Valor: yytext});return ':';}
"."                               {data.tokens.push({Tipo: 'PUNTO', Valor: yytext});return '.';}
","                               {data.tokens.push({Tipo: 'COMA', Valor: yytext});return ',';}
"--"                             {data.tokens.push({Tipo: 'GUION_GUION', Valor: yytext});return '--';}
"- "                             {data.tokens.push({Tipo: 'GUION', Valor: yytext});return '-';}
"+"                             {data.tokens.push({Tipo: 'MAS', Valor: yytext});return '+';}
%"                               {data.tokens.push({Tipo: 'PORCENTAJE', Valor: yytext});return '%';}
^"                               {data.tokens.push({Tipo: 'ELEVADO', Valor: yytext});return '^';}
@"                               {data.tokens.push({Tipo: 'ARROBA', Valor: yytext});return '@';}

"="                               {data.tokens.push({Tipo: 'IGUAL', Valor: yytext});return '=';}

"("                               {data.tokens.push({Tipo: 'PARENTESIS_IZQ', Valor: yytext});return '(';}
")"                               {data.tokens.push({Tipo: 'PARENTESIS_DER', Valor: yytext});return ')';}

// Palabras reservadas
"int"                             {data.tokens.push({Tipo: 'RESERVADA INT', Valor: yytext});return 'INT';}
"double"                         {data.tokens.push({Tipo: 'RESERVADA DOUBLE', Valor: yytext});return 'DOUBLE';}
"boolean"                       {data.tokens.push({Tipo: 'RESERVADA BOOLEAN', Valor: yytext});return 'BOOLEAN';}
"varchar"                      {data.tokens.push({Tipo: 'RESERVADA VARCHAR', Valor: yytext});return 'VARCHAR';}
"date"                         {data.tokens.push({Tipo: 'RESERVADA DATE', Valor: yytext});return 'DATE';}

"declare"                       {data.tokens.push({Tipo: 'RESERVADA DECLARE', Valor: yytext});return 'DECLARE';}
```

Aquí en el archivo json definimos las expresiones regulares que vamos a usar, inicializamos las variables e importamos código

```

%start ini

%%

ini : instructions EOF                                { return $1; }
;

instructions : instructions instruction                { $1.push($2); $$ = $1; }
            | instruction                            { $$ = $1 === null ? [] : [$1]; }
;

instruction : 'DECLARE' declaraciones ';'              { $$ = new Declaraciones($2, @1.first_line, @1.first_column); }
            | set ';'                                { $$ = $1; }
            | create_table ';'                        { $$ = $1; }
            | alter_table ';'                        { $$ = $1; }
            | drop ';'                                { $$ = $1; }
            | insert ';'                             { $$ = $1; }
            | select ';'                             { $$ = $1; }
            | error {
              data.errores.push({ line: this._$.first_line, column: this._$.first_column, type: 'Sintáctico', message: 'Error sintáctico, token no esperado '${yytext}' .' });
              console.log({ line: this._$.first_line, column: this._$.first_column, type: 'Sintáctico', message: 'Error sintáctico, token no esperado '${yytext}' .' });
            }
;

// -----Declaración y asignación de variables-----
declaraciones : declaraciones ',' declaracion { $1.push($3); $$ = $1; }
              | declaracion { $$ = [$1]; }
;
declaracion : '@' TK_IDENTIFIER tipodato { $$ = new DeclareNull($2, $3, @1.first_line, @1.first_column); }
            | '@' TK_IDENTIFIER tipodato 'DEFAULT'expression { $$ = new Declare($2, $3, $5, @1.first_line, @1.first_column); }
;

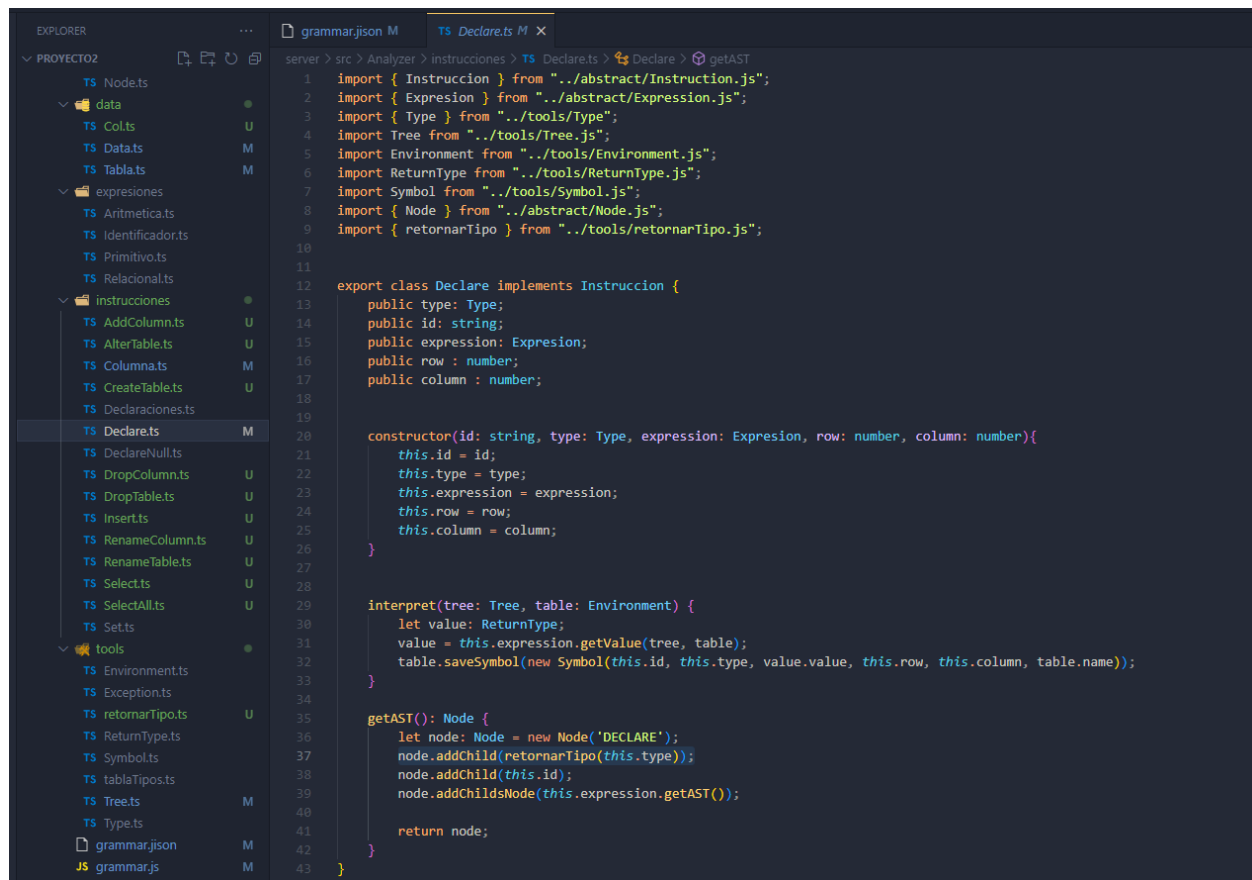
// ----- SET -----
set : 'SET' '@' TK_IDENTIFIER '=' expression { $$ = new Set($3, $5); }
;

//DDL
create_table : 'CREATE' 'TABLE' TK_IDENTIFIER '(' columns ')' { $$ = new CreateTable($3, $5, @1.first_line, @1.first_column); }
;
columns : columns ',' column { $1.push($3); $$ = $1; }
        | column { $$ = [$1]; }
;
column : TK_IDENTIFIER tipodato { $$ = new Columna($1, $2, @1.first_line, @1.first_column); }
;

alter_table : 'ALTER' 'TABLE' TK_IDENTIFIER accion_alter { $$ = new AlterTable($3, $4, @1.first_line, @1.first_column); }
;

```

Definimos las producciones que vamos a usar para nuestro analizador



```
server > src > Analyzer > instrucciones > TS Declare.ts > Declare > getAST
1  import { Instruccion } from "../abstract/Instruction.js";
2  import { Expresion } from "../abstract/Expression.js";
3  import { Type } from "../tools/Type";
4  import Tree from "../tools/Tree.js";
5  import Environment from "../tools/Environment.js";
6  import ReturnTipo from "../tools/ReturnTipo.js";
7  import Symbol from "../tools/Symbol.js";
8  import { Node } from "../abstract/Node.js";
9  import { retornarTipo } from "../tools/retornarTipo.js";
10
11
12  export class Declare implements Instruccion {
13      public type: Type;
14      public id: string;
15      public expression: Expresion;
16      public row : number;
17      public column : number;
18
19
20      constructor(id: string, type: Type, expression: Expresion, row: number, column: number){
21          this.id = id;
22          this.type = type;
23          this.expression = expression;
24          this.row = row;
25          this.column = column;
26      }
27
28
29      interpret(tree: Tree, table: Environment) {
30          let value: ReturnTipo;
31          value = this.expression.getValue(tree, table);
32          table.saveSymbol(new Symbol(this.id, this.type, value.value, this.row, this.column, table.name));
33      }
34
35      getAST(): Node {
36          let node: Node = new Node('DECLARE');
37          node.addChild(retornarTipo(this.type));
38          node.addChild(this.id);
39          node.addChildsNode(this.expression.getAST());
40
41          return node;
42      }
43  }
```

Usando el patron interprete, usamos clases para representar a los terminales y no terminales facilitando el escribir código, en esta clase implementamos la interfaz instruccion y generamos el ast

```
grammar.json M TS Expression.ts X
server > src > Analyzer > abstract > TS Expression.ts > ...
1  import Environment from "../tools/Environment";
2  import ReturnType from "../tools/ReturnType";
3  import Tree from "../tools/Tree";
4
5
6  export interface Expression{
7      row: number;
8      column: number;
9      getValue(tree: Tree, table: Environment): ReturnType;
10     getAST(): any;
11 }
```

Interfaz expresión que se encarga de obtener el valor y generar el ast, las implementamos mas adelante en cada clase respectiva

```
1  import Environment from "../tools/Environment";
2  import Tree from "../tools/Tree";
3  import { Node } from "../Node";
4
5  export interface Instruccion {
6      row: number;
7      column: number;
8
9      ⚡ interpret(tree: Tree, table: Environment): any;
10     getAST(): Node;
11 }
```

La interfaz Instrucción interpreta las instrucciones e igualmente genera el ast

```

import Symbol from "../Symbol.js";

export default class Environment {

  public name: string;
  public table: Map<string, Symbol>;
  public prev: Environment | undefined;

  constructor(prev?: Environment, name: string = "Global") {
    this.name = name;
    this.prev = prev;
    this.table = new Map<string, Symbol>();
  }

  public saveSymbol(symbol: Symbol) {
    if (!this.getSymbol(symbol.id)){
      symbol.environment = this.name;
      this.table.set(symbol.id, symbol);
    }
    return undefined;
  }

  public getSymbol(id: string) {
    let currentTable: Environment | undefined = this;

    while (currentTable != undefined) {
      if (currentTable.table.has(id)) {
        return currentTable.table.get(id);
      }

      currentTable = currentTable.prev;
    }

    return undefined;
  }

  public updateSymbol(symbol: Symbol) {
    let currentTable: Environment | undefined = this;

    while (currentTable != undefined) {
      if (currentTable.table.has(symbol.id)) {
        for (let entry of Array.from(currentTable.table)) {
          if (entry[0] === symbol.id) {
            entry[1].value = symbol.value;
          }
        }
      }
      currentTable = currentTable.prev;
    }
  }
}

```

Una clase environment en donde se encarga de ver el scope de las variables e instrucciones declaradas

```

server > src > Analyzer > tools > TS Trees > Tree > constructor
3 import Exception from './Exception.js';
4
5 export default class Tree {
6
7     public instructions: Array<any>;
8     public errors: Array<Exception>;
9     public console: string;
10    public globalTable: Environment | undefined;
11    public dot: string;
12    private count: number;
13    public tokens: Array<any>;
14
15
16    constructor(instructions: Array<any>) {
17        this.instructions = instructions;
18        this.console = '';
19        this.errors = [];
20        this.dot = '';
21        this.count = 0;
22        this.tokens = [];
23    }
24
25    public updateConsole(input: string) {
26        this.console += `${input}\n`;
27    }
28
29    public getDot(root: Node, flag: boolean = true) { //change Late
30        this.dot = '';
31        this.dot += `digraph {\nranksep=${flag ? 2 : 1};\nbgcolor = "#000010";\nedgecolor="#56cdff";\nnode [style="filled" fillcolor = "#0f111a" fontcolor = "white" color = "#007ac`;
32        this.dot += `n0[label="${root.value.replace('"', '\\\\"')}"];\n`;
33        this.count = 1;
34        this.travelCst("n0", root);
35        this.dot += " ";
36        return this.dot;
37    }
38
39    public travelCst(idRoot: any, nodeRoot: Node) {
40        for(let item of nodeRoot.children){
41            let name_child = `n${this.count}`;
42            this.dot += `${name_child} [label = "${item.value.replace('"', '\\\\"')}"];\n`;
43            this.dot += `${idRoot} -> ${name_child};\n`;
44            this.count++;
45            this.travelCst(name_child, item);
46        }
47    }
48
49 }

```

La clase tree una de las mas importantes para generar el ast aquí podemos generar nodo por nodo el código dot

```

3
4 const interpret = (bufferStrem: string): outParse => {
5     const data = Data.getInstance();
6     let tree: Tree | null;
7     let globalTable: Environment | null;
8
9     let instructions: Array<Instruccion>;
10    instructions = grammar.parse(bufferStrem);
11
12    tree = new Tree(instructions);
13    globalTable = new Environment(undefined, undefined);
14    tree.globalTable = globalTable;
15    for (let instr of instructions) {
16        try{
17            instr.interpret(tree, globalTable);}
18        catch(error){
19            console.log(error);
20        }
21    }
22    let rootAst: Node = new Node("Root");
23    let value: Node = new Node("Instructions");
24
25    for (let item of tree.instructions) {
26        try{
27            value.addChildsNode(item.getAST());}
28        catch(error){
29            console.log(error);
30        }
31    }
32
33    rootAst.addChildsNode(value);
34
35    let ast = tree.getDot(rootAst, false);
36    return {
37        "console": tree.console,
38        "ast": ast,
39        "reporte_tokens": crearTablaHTMLTokens(data.tokens),
40        "reporteErrores": crearTablaHTMLErrores(data.errores),
41        "reporte_simbolos": "test"
42    }
43

```

En esta clase unificamos todo el trabajo y lo enviamos por medio de una api en node js