MFMSCRIPT – PROYECTO 2

MANUAL TÉCNICO

Allen RománPROGRAMA REALIZADO POR EL ALUMNO ALLEN GIANKARLO ROMÁN VÁSQUEZ CARNET 202004745 PARA EL CURSO ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1 DE LA ESCUELA DE CIENCIAS Y SISTEMAS DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD SAN CARLOS DE GUATEMALA

Programa Utilizado

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Lenguaje Utilizado

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Anders Hejlsberg, diseñador de C# y creador de Delphi y Turbo Pascal, ha trabajado en el desarrollo de TypeScript. TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno).

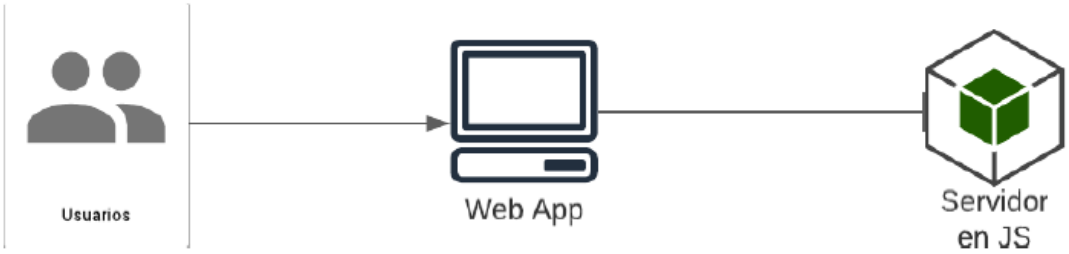
JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

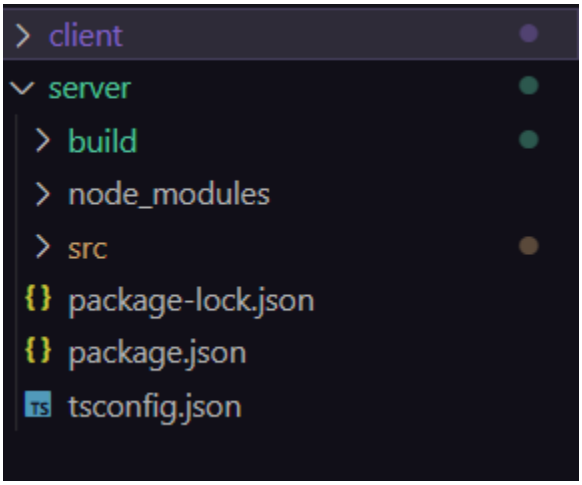
Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Código

Clases

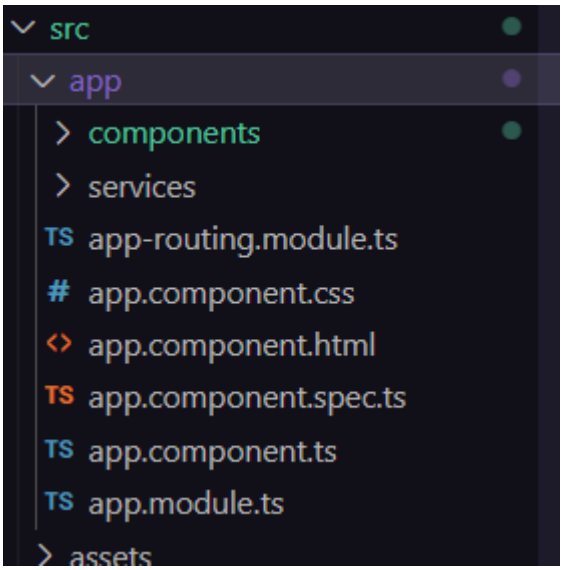
El programa cuenta con una arquitectura del tipo cliente servidor, la parte del ciente fue desarrollada con el framework y la parte del servidor con Typescript/JavaScript/Nodejs, el programa contiene por la parte del cliente el entorno gráfico en el que se pueden solicitar peticiones al servidor, en el servidor se hace análisis léxico y sintáctico con Jison como herramienta y análisis semántico con typescript que genera los archivos de javascript.





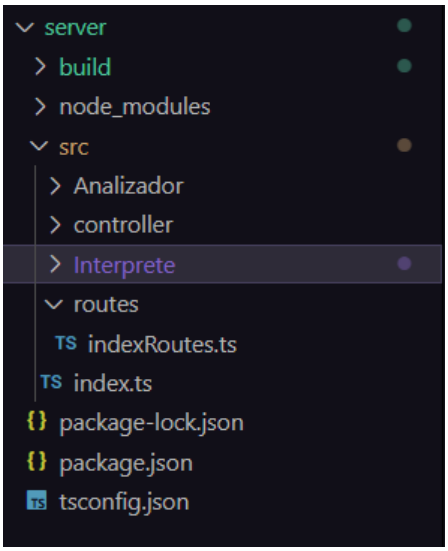
Interfaz gráfica - Client

La interfaz gráfica desarrollada en Angular cuenta con cinco components que se ejecutan, según la solicitud del cliente y con la ayuda de services se conecta el framework con el servidor.



Server

Los archivos Typescript son convertidos a Javascript para poder iniciar el servidor donde se reciben las solicitudes de parte de nuestro cliente y procesado por la api y devuletos.



Análisis léxico

En el server en la carpeta src/analizador se encuentra el archivo de jison con el análisis léxico que contiene las expresiones regulares, palabras reservadas que son utilizadas para el análisis léxico.

```
/* Ejemplo para la gramatica del interprete */

/* Definicion lexica */
%lex
%options case-insensitive

//Expresiones regulares
num [0-9]+
id      [a-zñA-ZÑ][a-zñA-ZÑ0-9_]*

//--> Cadena
escapechar      [\'\"\\ntr]
escape          \\{escapechar}
aceptacion      [^\"\\]
cadena          (\"({escape} | {aceptacion})*\")

//--> Caracter
escapechar2      [\'\"\\ntr]
escape2          \\{escapechar2}
aceptacion2      [^\'\\]
caracter         (\'({escape2} | {aceptacion2})\')

%%

/* Comentarios */
"//".*              {/*Ignoramos los comentarios simples*/}   id++
"/*"((\*+[^/*])|([^*]))*\**"*/"     {/*Ignorar comentarios con multiples lneas*/}
"tocharArray" { console.log("Reconocio : "+ yytext); return 'CHARARRAY'}
"new"  { console.log("Reconocio : "+ yytext); return 'NEW'}
"(int)"  { console.log("Reconocio : "+ yytext); return 'CASTEOINT'}
"(double)"  { console.log("Reconocio : "+ yytext); return 'CASTEODOUBLE'}
"(boolean)" { console.log("Reconocio : "+ yytext); return 'CASTEOBOOLEAN'}
"toString"  { console.log("Reconocio : "+ yytext); return 'CASTEOSTRING'}
"(string)"  { console.log("Reconocio : "+ yytext); return 'CASTEOSTRINGAUX'}
```

Análisis Sintáctico

En el server en la carpeta src/analizador se encuentra el archivo de jison que contiene la gramática libre de contexto que a su vez genera el AST con todos los terminales, no terminales, producciones y símbolo inicial.

```
instrucciones : instrucciones instruccion   { $$ = $1; $$.push($2); }
              | instruccion                 { $$ = new Array(); $$.push($1); }
              ;

instruccion : declaracion   { $$ =  $1; }
            | startwith      { $$ = $1; }
            | writeline     { $$ = $1; }
            | asignacion    { $$ = $1; }
            | sent_if       { $$ = $1; }
            | sent_while    { $$ = $1; }
            | sent_Dowhile  { $$ = $1; }
            | BREAK PYC     { $$ = new detener.default(); }
            | sent_switch   { $$ = $1; }
            | sent_for       { $$ = $1; }
            |  ID DECRE PYC  { $$ = new asignacion.default($1, new aritmetica.default(new identificador.default($1, @1.first_line, @1.last_column
            | ID INCRE  PYC { $$ = new asignacion.default($1, new aritmetica.default(new identificador.default($1, @1.first_line, @1.last_column)
            | CONTINUE PYC  { $$ = new continuar.default(); }
            | funciones     { $$ = $1; }
            | llamada PYC   { $$ = $1; }
            | RETURN PYC        { $$ = new retorno.default(null); }
            | RETURN e PYC      { $$ = new retorno.default($2); }
            | error         { console.log("Error Sintactico: " + yytext
                                    + " linea: " + this._$.first_line
                                    + " columna: " + this._$.first_column);

                              new errores.default("Sintactico", "No se esperaba el caracter "+ yytext ,
                                            this._$.first_line ,this._$.first_column);
                            }
            ;
```

Análisis Semántico

En el server en la carpeta src/interprete se encuentra todos los archivos que llevan a cabo el análisis semántico de la aplicación, así como las interfaces que aplican ya sea las instrucciones o expresiones, además del ast y tabla de símbolos.

Repositorio: https://github.com/Allenrovas/-OLC1_PY2_202004745