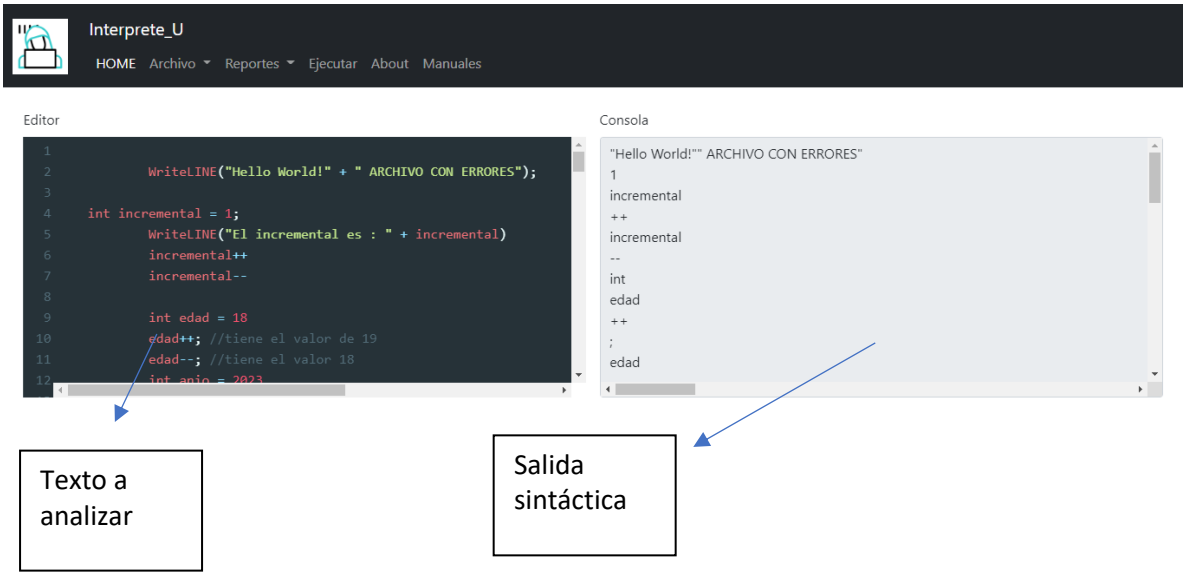
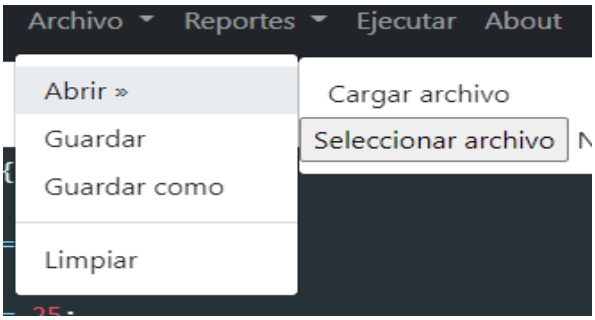
MANUAL DE TÉCNICO

Requisitos:

Sistema Operativo: Windows 10
RAM: 4GB
Espacio en disco: 80MB
Navegador Web: Google Chrome o cualquiera compatible.

INTERFAZ





Abrir: Abre un nuevo documento

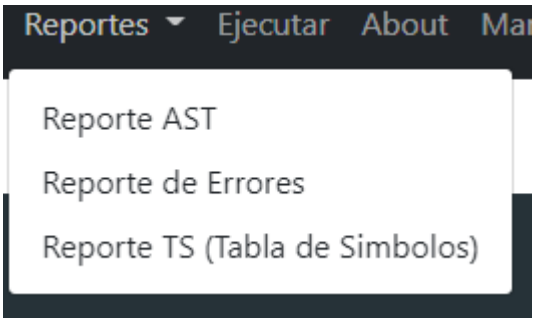
Seleccionar archivo: Permite seleccionar un archivo para subirlo en la aplicación.

Cargar archivo: Carga un archivo para mostrarlo en el área de texto a analizar.

Guardar: Guarda cambios en el documento

Guardar Como: Guarda un nuevo documento por primera vez

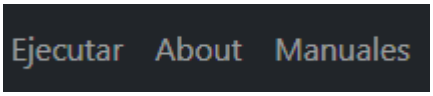
Limpiar: Limpia las áreas de texto a analizar y salida sintáctica.



Reporte TS (Tabla de Símbolos): Muestra una tabla de símbolos representando los ámbitos de las variables.

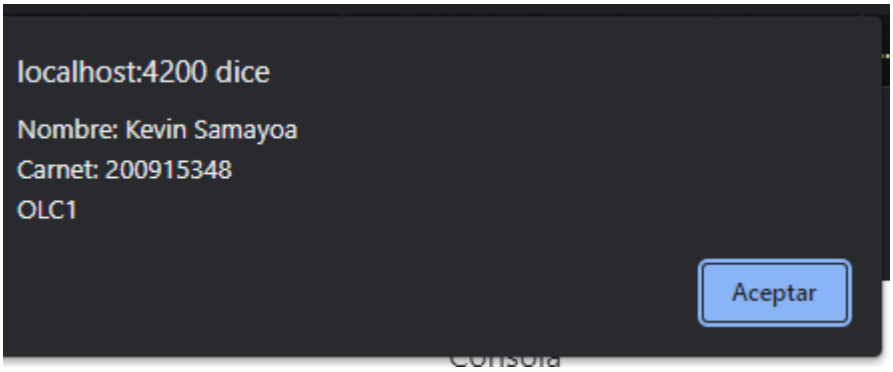
Reporte de Errores: Genera un reporte con los errores léxicos y sintácticos encontrados.

Reporte AST: Muestra una imagen del Árbol de Análisis Sintáctico



Ejecutar: Genera el análisis léxico y sintáctico de la entrada, mostrando la traducción en Python en la consola.

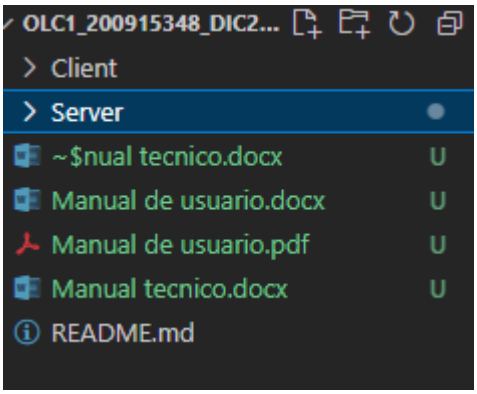
About: Muestra información sobre el desarrollador de la aplicación y los interesados (a quien se entrega la aplicación).



Manuales: Muestra los siguientes manuales y gramática del analizador sintáctico.

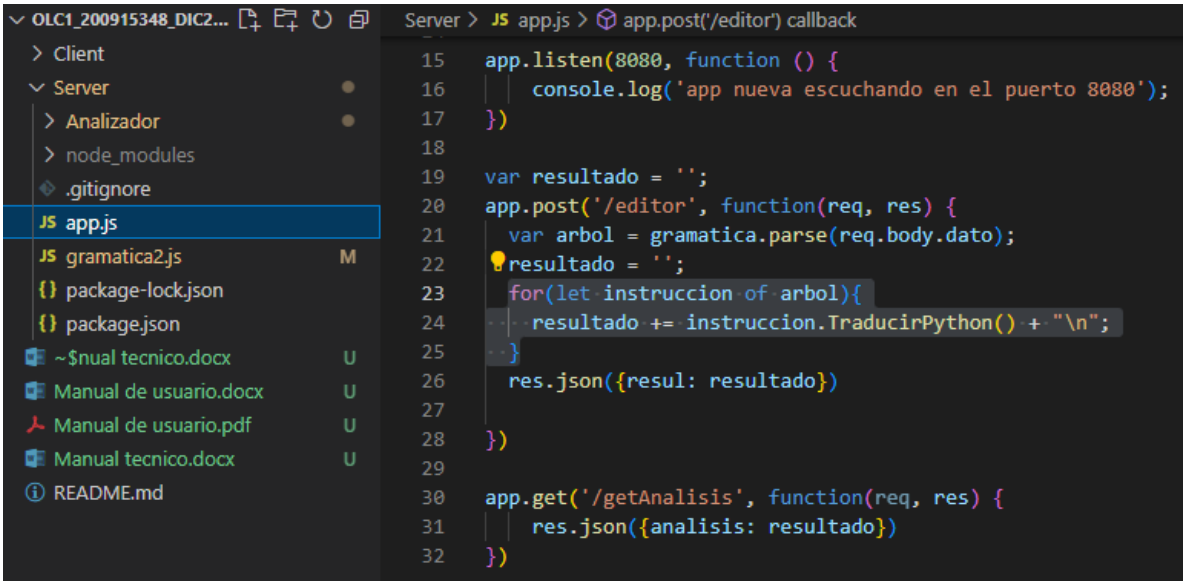
- Manual del Usuario: Muestra el manual de usuario desde la aplicación.
- Manual técnico: Muestra un manual con especificaciones del funcionamiento de la aplicación.
- Gramática: Muestra un archivo con la gramática utilizada para hacer el análisis sintáctico.

PROGRAMACIÓN



Client: Se maneja todo lo concerniente al usuario final, es decir, la interfaz grafica para el uso de dicho usuario.

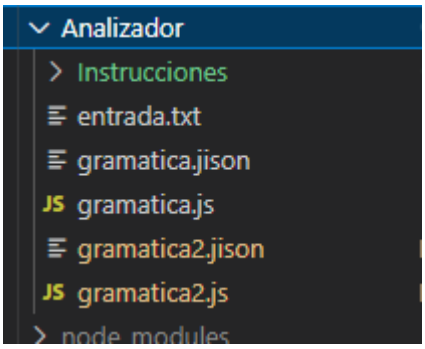
Server: es el back end de la aplicación, se encuentra el servidor y los analizadores léxico y sintáctico por medio de un patrón interprete para realizar la traducción de lenguaje C# a Python.



App.js: Contiene el servidor de la aplicación.

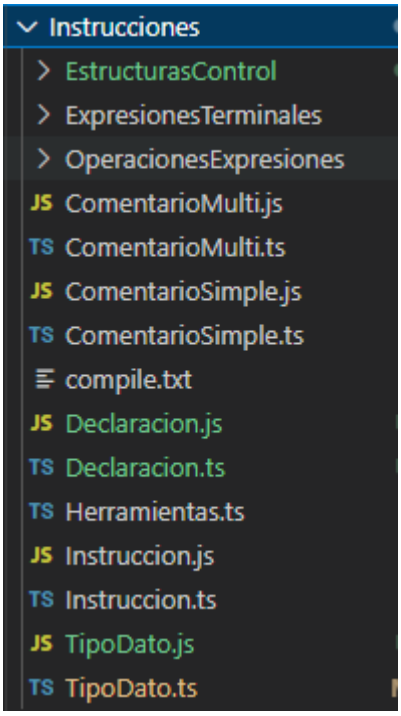
- App.post: método que realiza el análisis y guarda el resultado en la variable “resultado”.
- App.get: método que retorna el resultado del análisis para mostrarlo en la consola.

Analizador: Carpeta que contiene el analizador.



Gramatica2.jison: Es la gramática utilizada para el análisis

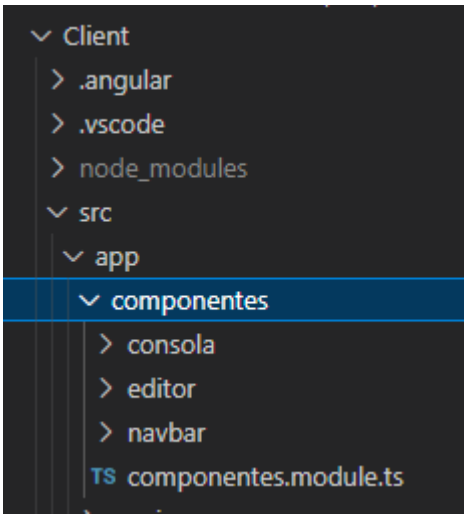
Instrucciones: Carpeta que contiene las clases del patrón interprete para realizar el análisis



EstructurasControl: Carpeta que contiene los archivos que definen y traducen las estructuras de control (if, for, while, switch, doWhile).

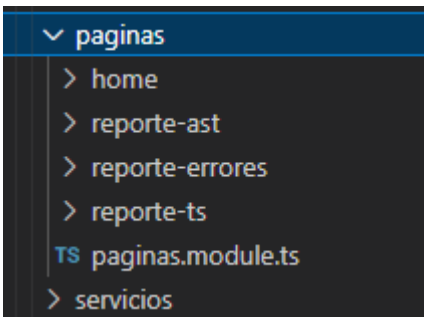
ExpresionesTerminales: Carpeta que contine las clases para traducir expresiones consideradas terminales en la gramática (cadena, numero, etc).

OperacionesExpresiones: Carpeta que contiene las clases que traducen operaciones entre valores, como operaciones aritméticas, lógicas, relacionales y booleanos.



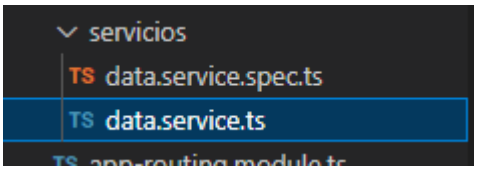
La interfaz fue creada utilizando angular, a continuación, se describen los componentes utilizados en esta aplicación.

- Consola: Se usa para mostrar el resultado del análisis y traducción.
- Editor: Se usa para crear el área de texto donde se capturan las instrucciones para el análisis.
- Navbar: crea una barra de navegación que incluye los botones que realizan todas las funcionalidades de la aplicación.



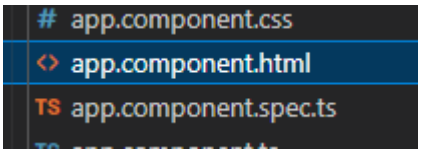
Paginas: Incluye las paginas en las que puede navegar el usuario

- Home: contiene el área de edición del texto para analizar y la consola para mostrar los resultados.
- Reporte-ast: pagina que muestra el reporte del árbol de análisis sintáctico.
- Reporte-errores: pagina que muestra un reporte con los errores léxicos y sintácticos encontrados.
- Reporte-ts: pagina para mostrar el reporte de tabla de símbolos.




```typescript
export class DataService {

  contenidoEditor$ = new EventEmitter<string>();
  contenidoConsola$ = new EventEmitter<string>();
  limpiar$ = new EventEmitter<string>();

  //Para conectar al servidor (Express)
  URL = "http://localhost:8080"

  //constructor() { }
  constructor( private http:HttpClient ) { }

  //Metodos para usar con el servidor
  getAnalizador(json: any){
    return this.http.post(`${this.URL}/editor`, json);
  }

  getAnalisis(){
    return this.http.get(`${this.URL}/getAnalisis`);
  }
```

Servicios -> data.service: Contiene los servicios utilizados para conectar la aplicación con el servidor.



En app.component.html es el apartado que ejecuta toda la interfaz para el usuario.

GRAMATICA

```
INIT
    :INSTRUCCIONES EOF
;

INSTRUCCIONES
    :INSTRUCCIONES INSTRUCCION
    |INSTRUCCION

;

INSTRUCCION
    :DECLARACION
    |ASIGNACION
    |IF
    |WHILE
    |SWITCH
    |DO
    |FOR
    |BREAK
    |CONTINUE
    |RETURN
    |PRINT
    |METODO
    |FUNCION
    |LLAMADAFUNCION PComa
    |VECTORES
    |LLAMADAVECTOR PComa
    |error
;

DECLARACION
    :TIPO LISTAVARIABLES Igual EXPRESION PComa
    |TIPO LISTAVARIABLES PComa
;

ASIGNACION
    :LISTAVARIABLES Igual EXPRESION PComa
;

LISTAVARIABLES
    :LISTAVARIABLES Coma Id
    |Id
;
```

```
TIPO
    :resString
    |resChar
    |resBool
    |resInt
    |resDouble
;

EXPRESION
    //ARITMETICOS
    : EXPRESION 'Mas' EXPRESION
    | EXPRESION 'Menos' EXPRESION
    | EXPRESION 'Por' EXPRESION
    | EXPRESION 'Div' EXPRESION
    //RELACIONALES
    | EXPRESION 'MayorQue' EXPRESION
    | EXPRESION 'MenorQue' EXPRESION
    | EXPRESION 'Igualdad' EXPRESION
    | EXPRESION 'Distinto' EXPRESION
    | EXPRESION 'MayorIgual' EXPRESION
    | EXPRESION 'MenorIgual' EXPRESION
    //LOGICOS
    | EXPRESION 'And' EXPRESION
    | EXPRESION 'Or' EXPRESION
    | 'Not' EXPRESION
    //UNARIOS
    | 'Menos' EXPRESION %prec UMenos
    //AGRUPACION
    | ParA EXPRESION ParC
    //TERNARIO
    | EXPRESION Interrogacion EXPRESION DosPuntos EXPRESION
    //TERMINALES
    | Cadena
    | Caracter
    | Entero
    | Decimal
    | Verdadero
    | Falso
    | LLAMADAFUNCION
    | LLAMADAVECTOR
    | Id
;


INCREMENTALES
```

```
    :Id Incremento
    |Id Decremento
;

IF
    :resIf ParA EXPRESION ParC LlaveA INSTRUCCIONES LlaveC ELSE
;

ELSE
    :resElse resIf ParA EXPRESION ParC LlaveA INSTRUCCIONES LlaveC ELSE
    |resElse LlaveA INSTRUCCIONES LlaveC
    |
;

WHILE
    :resWhile ParA EXPRESION ParC LlaveA INSTRUCCIONES LlaveC
;

SWITCH
    :resSwitch ParA EXPRESION ParC LlaveA CASE LlaveC
;

CASE
    :resCase Caracter DosPuntos INSTRUCCIONES CASE
    |resCase Cadena DosPuntos INSTRUCCIONES CASE
    |resCase Decimal DosPuntos INSTRUCCIONES CASE
    |resCase Entero DosPuntos INSTRUCCIONES CASE
    |resDefault DosPuntos INSTRUCCIONES
;

DO
    :resDo LlaveA INSTRUCCIONES LlaveC resWhile ParA EXPRESION ParC PComa
;

FOR
    :resFor ParA TIPO Id Igual EXPRESION PComa EXPRESION PComa INCREMENTALES
ParC LlaveA INSTRUCCIONES LlaveC
;

BREAK
    :resBreak PComa
;


CONTINUE
```

```
    :resContinue PComa
;

RETURN
    :resReturn EXPRESION PComa
    |resReturn PComa
;

PRINT
    :resWrite ParA EXPRESION ParC PComa
;

METODO
    :resVoid Id ParA ParC LlaveA INSTRUCCIONES LlaveC
    |resVoid Id ParA PARAMETRO ParC LlaveA INSTRUCCIONES LlaveC
;

FUNCION
    :TIPO Id ParA ParC LlaveA INSTRUCCIONES LlaveC
    |TIPO Id ParA PARAMETRO ParC LlaveA INSTRUCCIONES LlaveC
;

PARAMETRO
    :PARAMETRO Coma TIPO Id
    |TIPO Id
;

LLAMADAFUNCION
    :Id ParA ParC
    |Id ParA LISTAEXPRESION ParC
;

LISTAEXPRESION
    :LISTAEXPRESION Coma EXPRESION
    |EXPRESION
;

VECTORES
    :TIPO Id CorA CorC Igual resNew TIPO CorA EXPRESION CorC PComa
    |TIPO Id CorA CorC Igual LlaveA LISTAEXPRESION LlaveC PComa
;

LLAMADAVECTOR
    :Id CorA EXPRESION CorC
;
```