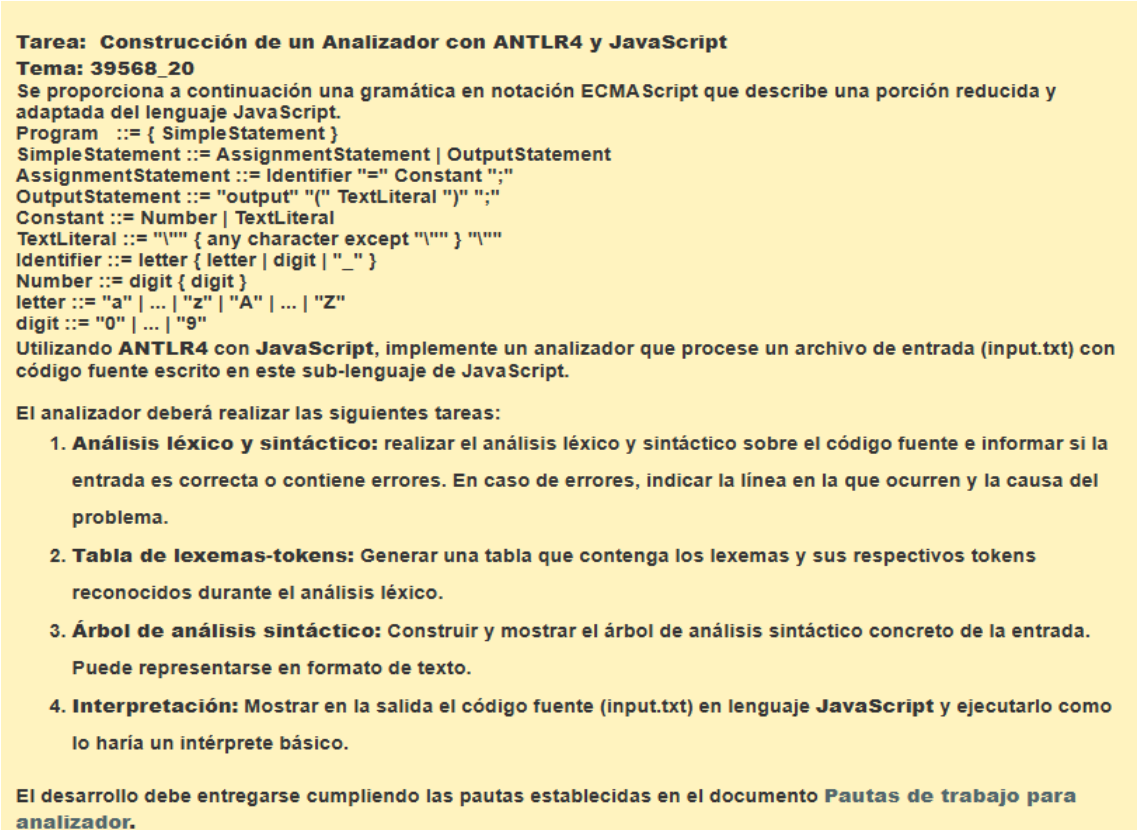
Gramática Asignada - Analizador Sintáctico

Santiago Zingaretti - Legajo 51163

A continuación se adjunta una captura con la gramática que se asignó para este trabajo y más abajo la gramática en texto (Recordar que también está adjunto el archivo en formato .txt bajo el nombre de “Gramática Asignada.txt”)



Tarea: Construcción de un Analizador con ANTLR4 y JavaScript
Tema: 39568_20
Se proporciona a continuación una gramática en notación ECMAScript que describe una porción reducida y adaptada del lenguaje JavaScript.

```
Program   ::= { SimpleStatement }
SimpleStatement ::= AssignmentStatement | OutputStatement
AssignmentStatement ::= Identifier "=" Constant ";"
OutputStatement ::= "output" "(" TextLiteral ")" ";"
Constant ::= Number | TextLiteral
TextLiteral ::= "\"" { any character except "\"" } "\""
Identifier ::= letter { letter | digit | "_" }
Number ::= digit { digit }
letter ::= "a" | ... | "z" | "A" | ... | "Z"
digit ::= "0" | ... | "9"
```

Utilizando **ANTLR4** con **JavaScript**, implemente un analizador que procese un archivo de entrada (input.txt) con código fuente escrito en este sub-lenguaje de JavaScript.

El analizador deberá realizar las siguientes tareas:

1. **Análisis léxico y sintáctico:** realizar el análisis léxico y sintáctico sobre el código fuente e informar si la entrada es correcta o contiene errores. En caso de errores, indicar la línea en la que ocurren y la causa del problema.
2. **Tabla de lexemas-tokens:** Generar una tabla que contenga los lexemas y sus respectivos tokens reconocidos durante el análisis léxico.
3. **Árbol de análisis sintáctico:** Construir y mostrar el árbol de análisis sintáctico concreto de la entrada. Puede representarse en formato de texto.
4. **Interpretación:** Mostrar en la salida el código fuente (input.txt) en lenguaje **JavaScript** y ejecutarlo como lo haría un intérprete básico.

El desarrollo debe entregarse cumpliendo las pautas establecidas en el documento **Pautas de trabajo para analizador.**

Gramática pasada a texto:

Program ::= { SimpleStatement }

SimpleStatement ::= AssignmentStatement | OutputStatement

AssignmentStatement ::= Identifier "=" Constant ";"

OutputStatement ::= "output" "(" TextLiteral ")" ";"

Constant ::= Number | TextLiteral

TextLiteral ::= "\"" { any character except "\"" } "\""

Identifier ::= letter { letter | digit | "_" }

Number ::= digit { digit }

letter ::= "a" | ... | "z" | "A" | ... | "Z"

digit ::= "0" | ... | "9"