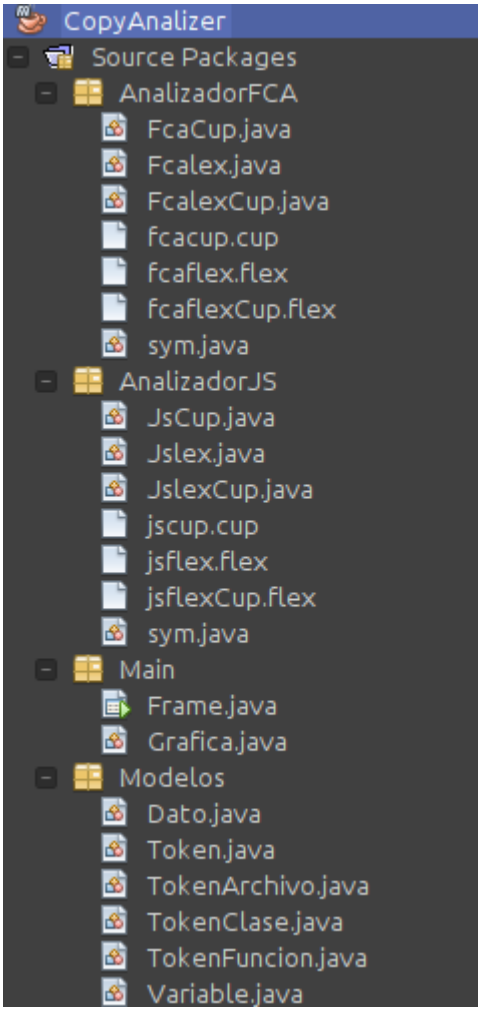
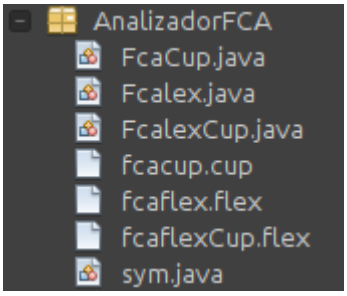
Manual Técnico

Nombre: Elder Andrade **Carnet:** 201700858

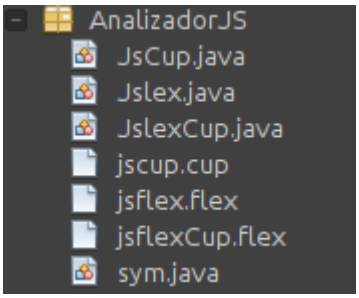


CopyAnalizer es un programa para calcular la magnitud de copia entre dos archivos se compone de los siguientes paquetes y cada uno con sus clases y archivos que conforman su funcionamiento.

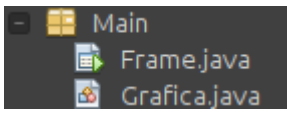
Paquete AnalizadorFCA: Utiliza lo que es el programa JFLEX y JCUP para analizar las reglas léxicas y sintácticas del lenguaje creado FCA, esto ejecuta un reporte estadístico que genera graficas y archivos JSON y Html que componen toda la información de los archivos a utilizar.



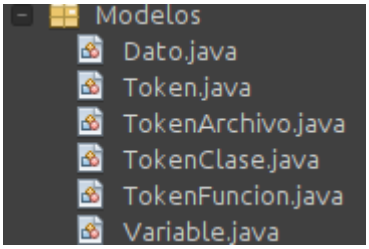
Paquete AnalizadorJS: Utiliza lo que es el programa JFLEX y JCUP para analizar las reglas léxicas y sintácticas del lenguaje JS, esto ejecuta un comparador estadistico que genera graficas y archivos JSON y Html que compone toda la información de los archivos a utilizar.



Paquete Main: Contiene toda la funcionalidad y las característica estéticas que el programa necesita.



Paquete Modelos: contiene todas las estructuras de datos utilizadas en el transcurso del proyecto.



Gramática Utilizada.

<terminador> ::= “;”
<coma> ::= “,”
<comparador> ::= “>” | “<” | <operador_asignación> <operador_asignación> | “>=” | “<=”
<operador_lógico> ::= “&&” | “||”
<operador_aritmético> ::= “+” | “-“ | “” | “/”*
<operador_asignación> ::= “=”
<llave_inicio> ::= “{“
<llave_fin> ::= “}“
<paréntesis_inicio> ::= “(“
< paréntesis _fin> ::= “)“
<var> ::= “var“
<return> ::= “return“
<if> ::= “if“
<else> ::= “else“
<while> ::= “while“
<function> ::= “function“
<programa>::= <programa> <sentencia>
|<programa> <programa>
|<sentencia>
|<sentencia>::=
|<declaración_funcion>
|<declaración_var> <terminador>
|<declaración_var2> <terminador>
|<asignación> <terminador>
|<expresión> <terminador>
|<setencia_return> <terminador>
|<estructura>
<declaración_var>::=
|<var> <identificador>
|<declaración_var> <coma> <identificador>
|<declaración_var2> <coma> <identificador>
<declaración_var1>::=
|<var> <asignación1>
|<declaración_var> <coma> <asignación1>
|<declaración_var2> <coma> <asignación1>
<declaración_var2>::=
|<declaración_var1> <expresión>
|<declaración_var2> <op_aritmético> <expresión>
<asignación>::=
|<asignación1> <expresión>
|<asignación> <op_aritmético> <expresión>
<asignación1> ::= <identificador> <op_asignación>
<declaración_función> ::= <function> <identificador> <paren_inicio> <paren_fin> <bloque>

|<declaración_función1> <paren_fin > <bloque>
<declaración_función1>::=
|<function> <identificador> <paren_inicio> <identificador>
|<declaración_función1> <declaración_función1> <coma> <identificador>
<bloque>::=
|<llave_inicio> <programa> <llave_fin>
|<llave_inicio> <llave_fin>
<expresión> ::=
|<paren_inicio> <paren_fin>
|<expresión> <op_aritmético> <expresión>
|<término>
<término>::=
|<identificador>
|<llamada>
|<condición>
<conparación>
<literal>
<condición> ::= <expresión> <op_lógico> <expresión>
<comparación> ::= <expresión> <comparador > <expresión>
<sentencia_return> ::= <return> <expresión>
|<sentencia_return> <op_aritmético> <expresión>
<llamada>::=
|<identificador> <paren_inicio> <paren_fin>
|<llamada1> <paren_fin>
<llamada1>::=
|<identificador> <paren_inicio> <expresión>
|<llamada1> <op_aritmético> <expresión>
|<llamada1> <coma> <expresión>
<estructura>::=
|<estructura_if>
|<estructura_while>
<estructura_if>::=
|<estructura_if1> <llave_fin>
|<estructura_if1> <llave_fin> <else> <bloque>
<estructura_if1> ::=
|<if> <expresión> <llave_inicio> <programa>
|<estructura_if1> <programa>
<estructura_while> ::= <estructura_while1> <llave_fin>
<estructura_while1> ::= <while> <expresión> <llave_inicio> <programa>
|<estructura_while1> <programa>
<literal>::=
|<entero>
|<flotante>
|<cadena>
<identificador> = (<letra>)+ (<letra>|<numero>|"_")*
<entero> ::= (<número>)+
<flotante> ::= <entero> . <entero>
<cadena> ::= "'" . (<letra> | <número> | <símbolo>)* . "'"
<letra> ::= "a" | "A" | "b" | "B" | ... | "z" | "Z"

<número> ::= “1”| “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” | “0”
<símbolo> ::= “-“ | “_” | “.” | “:” | “,” | “;” | “+” | “=” | “#” | “$” | “%” | “&” | “/” | “(“ | “)” | “>” | “<”

Bibliografía:

https://es.scribd.com/document/366609333/Analisis-Sintactico-del-lenguaje-JavaScript-utilizando-teoria-de-automatas

https://ericknavarro.io/