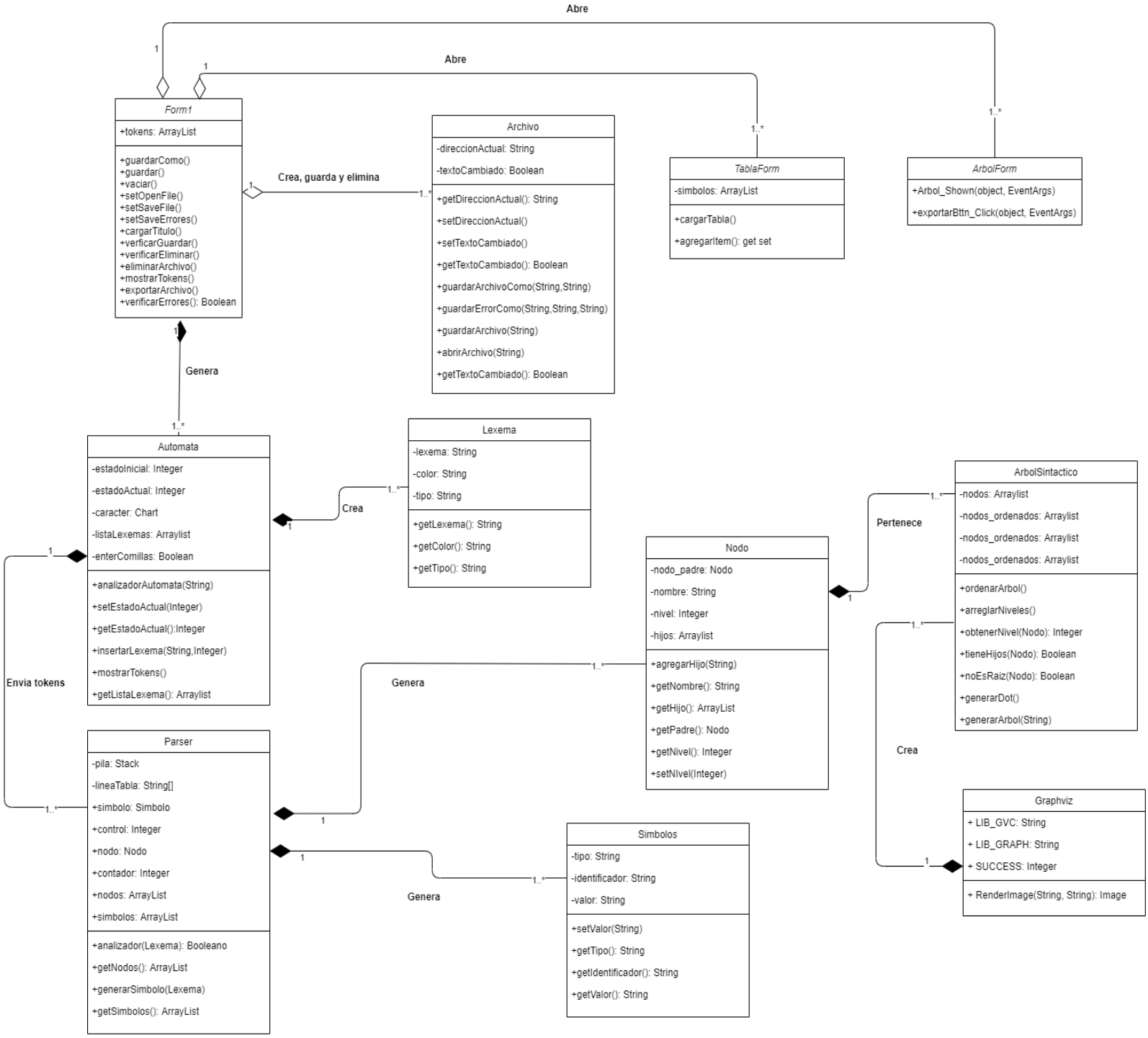
Manual Técnico

1. Diagrama de Clase



2. Expresión Regular

((entero) | (decimal) | (cadena) | (booleano) | (carácter) | ((-)(1-9)+(0-9)) | (0-9)+ | (-)(0-9)+(.)(0-9)+ | (0-9)+(.)(0-9)+ | (")((A-Z) | (a-z) | (0-9))*(") | (verdadero) | (falso) | ((a-z)|(A-Z)) | (')((a-z)|(A-Z))(') | (+) | (-) | (*) | (/) | (++) | (--) | (>) | (<) | (>=) | (<=) | (==) | (!=) | (||) | (&&) | (!) | (|) | (=) | (;) | (SI) | (SINO) | (SINO_SI) | (MIENTRAS) | (HACER) | (DESDE) | (HASTA) | (INCREMENTO) | (((//)((A-Z) | (a-z) | (0-9))*) | (((/*)((A-Z) | (a-z) | (0-9))*(*/)))*

3. Definición de Lenguaje

3.1 Lexemas aceptados:

a) Palabras Reservadas:
SI, SINO, SINO_SI, MIENTRAS, HACER, DESDE, HASTA, INCREMENTO, entero, decimal, cadena, booleano, carácter.

b) Identificadores:
Los identificadores deben llevar un guion bajo al principio.
Ejemplo: *_identificador.*

c) Tipos de datos: Entero, decimales positivos y negativos, cadenas con comillas, palabras verdadero y falso, caracteres de un 1 bit.

d) Operadores:
   a. Aritméticos: +, -, *, /, ++, --
   b. Relacionales: >, <, >=, <=, ==, !=
   c. Lógicos: ||, &&, !

e) Signos de Agrupación: (,)

f) Asignación: =

g) Fin de sentencia: ;

h) Comentarios
   a. // ejemplo 1
   b. /* ejemplo 2 */

i) Palabras Validas:
Las palabras validas serán todas aquellas que estén definidas dentro del lenguaje, otra que no exista será un error. Todo tipo de operador tendrá la

función de un separador, al igual que los espacios vacíos, delimitan hasta donde llegara nuestro token.
Ejemplo: 55+55+88 verdadero&&falso.

3.2 Sintaxis del lenguaje:

a) Estructura del código:
   Para que el código sea sintácticamente valido, será agrupado dentro de un método “principal”, así como se ejemplifica a continuación:

   principal(){
   // Sentencias
   }

b) Estructura de sentencias
   a. Fin de línea:
      Al final de cada línea de código se debe escribir punto y coma (“;”).

   b. Variables:
      Las variables podrán ser de tipo entero, decimal, cadena, booleano y carácter. Primero se define el tipo de dato, luego el identificador. Este puede ser inicializado con un valor o simplemente declarado. También se puede crear varias variables en una sola línea.

      Ejemplos:
      cadena _id = “cadena”;
      entero _id = 55;
      booleano _id;
      decimal _id1,_id2,_id3;

   c. “SI”, “SINO_SI” y “SINO”
      SI (“operadores condicionales”){
      //Sentencias
      } SINO_SI (“operadores condicionales”){
      //Sentencias
      } SINO {
      //Sentencias
      }

d. “MIENTRAS” y “HACER”:

MIENTRAS(“operadores condicionales”){
//Sentencias
}

HACER {
//Sentencias
} MIENTRAS (“operadores condicionales”)

e. “DESDE”:

DESDE _i = 0 HASTA _i < 20 INCREMENTO 1{
//Sentencias
}

f. “imprimir” y “leer”:
Imprimir podrá incluir uno o varios valores, estos siendo de tipo cadena, entero, decimal o identificador. Leer solo incluye un identificador/

imprimir(_id+_id1+55);
leer(_id);

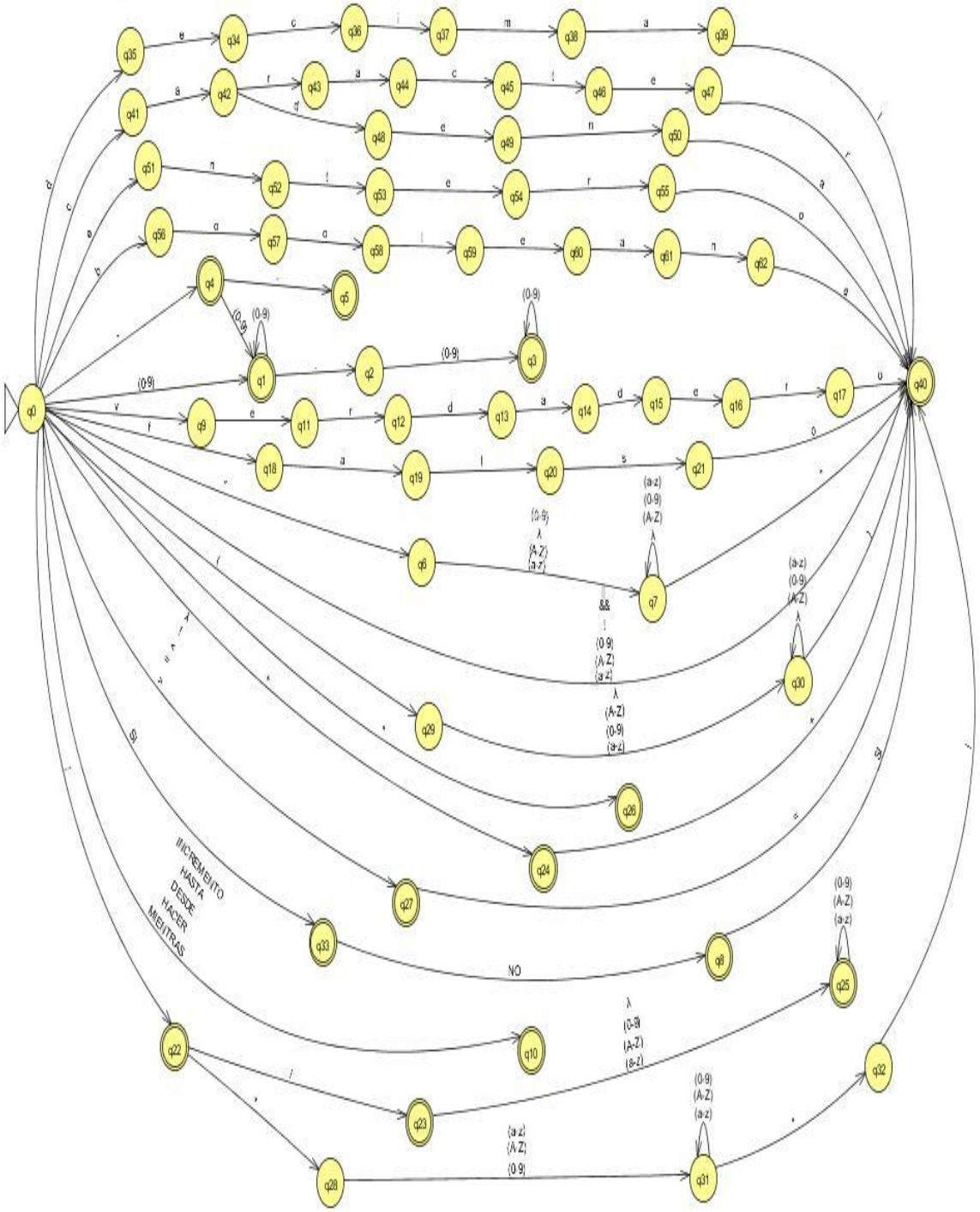
g. Igualación de variables:
A una variable se le puede asignar un valor mediante su identificador. El valor puede ser una cadena o concatenación de cadenas, un entero, decimal o una mezcla de estos en una operación matemática, booleano (verdadero o falso) o un carácter.

_id = 55;
*_id = (55*55) + 125;*
_id = “cadena 1” + ”cadena 2”;

h. Operadores condicionales:
Estos pueden ser uno o varios separados por “&&” o “||” y agrupados dentro de paréntesis.

(_id => 10) && (_id2 == 10) || (_id3 < 10) || (_id4 != 10)

4. Autómata



5. Analizador sintáctico descendente.

5.1 Gramática LL1:

A → principal {B

B → L}

// Produce líneas de código

L → D L

| G L

| F L

| Z L

| ℰ

// Produce una igualación de una variable o aumento

Z → ID J;

J → ++

| --

|= Z'

Z' → O

| CADENA O'

| BOOLEANO

| CARÁCTER

O → ENTERO N

| DECIMAL N

| *ID N*

| *(O) N*

N → *+ O*

| *- O*

| ** O*

| */ O*

| *ℰ*

O' → *+ N'*

| *ℰ*

N' → *CADENA O'*

| *ID O'*

//Produce la declaración de una variable

D → *D';*

D' → *decimal ID P*

| *entero ID Q*

| *booleano ID R*

| *cadena ID S*

| *carácter ID T*

P → *= O*

|*, I*

|*ℰ*

$Q \rightarrow = \ Q''$

$|, I$

$|\mathcal{E}$

$Q'' \rightarrow ENTERO\ Q'$

$|\ ID\ Q'$

$|\ (Q'')\ Q'$

$Q' \rightarrow + Q''$

$|\ - Q''$

$|\ * Q''$

$|\ / Q''$

$|\ \mathcal{E}$

$R \rightarrow = BOOLEANO$

$|, I$

$|\mathcal{E}$

$S \rightarrow = CADENA\ O'$

$|, I$

$|\mathcal{E}$

$T \rightarrow = CARACTER$

$|, I$

$|\mathcal{E}$

I → ID I'

I' → ,I

| ε

//Produce las funciones de leer e imprimir

G → imprimir (C

| leer (ID);

C → M);

M → CADENA M'

| ENTERO M'

| DECIMAL M'

| ID M'

M' → +M

| ε

//Produce condicionales

F → SI (V) S' E E'

| MIENTRAS (V) S'

| HACER S' MIENTRAS (V)

| DESDE ID = ENTERO HASTA ID S'' ENTERO INCREMENTO ENTERO S'

V → (V' X V') Q'''

| V' X V'

V' → *ID*

| *ENTERO*

| *DECIMAL*

| *BOOLEANO*

| *CADENA*

X → >

| <

| ==

| <=

| >=

|*!* =

Q''' → *&& V*

| || *V*

| *Ɛ*

S' → *{L}*

S'' → <

| >

|=

|<=

|>=

E → *SINO_SI(V) S' E*

| *Ɛ*

$E' \rightarrow SINO\ S'$

$\quad | \ \varepsilon$

5.2 Tabla de Primeros.

No Terminales	Primeros
P(A)	{ principal }
P(B)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID, Ɛ }
P(L)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID, Ɛ }
P(Z)	{ ID }
P(J)	{++,--,=}
P(Z')	{ ENTERO, DECIMAL, CADENA, BOOLEANO, CARÁCTER, ID, (}
P(O)	{ ENTERO, DECIMAL, ID, (}
P(N)	{ +, -, *, /,Ɛ }
P(O')	{ + , Ɛ }
P(N')	{ CADENA, ID }
P(D)	{ decimal, entero, booleano, cadena, carácter }
P(D')	{ decimal, entero, booleano, cadena, carácter }
P(P)	{ = , , , Ɛ }
P(Q)	{ = , , , Ɛ }
P(Q')	{ +, -, *, /, Ɛ }
P(Q'')	{ ENTERO, ID, (}
P(R)	{ = , , , Ɛ }
P(S)	{ = , , , Ɛ }
P(T)	{ = , , , Ɛ }
P(I)	{ ID }
P(I')	{ , , Ɛ}
P(G)	{imprimir, leer }
P(C)	{ CADENA, ENTERO, DECIMAL, ID }
P(M)	{ CADENA, ENTERO, DECIMAL, ID }
P(M')	{ +, Ɛ }
P(F)	{ SI, MIENTRAS, HACER, DESDE }
P(V)	{ (, ID, ENTERO, DECIMAL, BOOLEANO, CADENA }
P(V')	{ ID, ENTERO, DECIMAL, BOOLEANO, CADENA }
P(X)	{ >, <, ==, <=, >=, != }
P(Q''')	{ &&, ||,Ɛ }
P(S')	{ { }
P(S'')	{ >, <, = <=, >= }
P(E)	{ SINO_SI, Ɛ }
P(E')	{ SINO, Ɛ }

.

5.3 Tabla de Siguientes

No Terminal	Siguientes
S(A)	{ $ }
S(B)	{ $ }
S(L)	{ } }
S(Z)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(J)	{ ; }
S(Z')	{ ; }
S(O)	{ ; ,) }
S(N)	{ ; ,) }
S(O')	{ ; }
S(N')	{ ; }
S(D)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(D')	{ ; }
S(P)	{ ; }
S(Q)	{ ; }
S(Q')	{ ; ,)}
S(Q'')	{ ; ,)}
S(R)	{ ; }
S(S)	{ ; }
S(T)	{ ; }
S(I)	{ ; }
S(I')	{ ; }
S(G)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(C)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(M)	{), ; }
S(M')	{), ; }
S(F)	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(V)	{) }
S(V')	{ >, <, ==, <=, >=, != ,) }
S(X)	{ ID, ENTERO, DECIMAL, BOOLEANO, CADENA }
S(Q''')	{) }
S(S')	{ SINO_SI, Ɛ, decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(S'')	{ ENTERO }
S(E)	{ SINO, Ɛ , decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }
S(E')	{ decimal, entero, booleano, cadena, carácter, imprimir, leer, SI, MIENTRAS, HACER, DESDE, ID }

5.4 Tabla de Análisis sintáctico.

Se adjunta archivo Excel con la tabla al proyecto debido al tamaño de esta.