

Universidad de San Carlos de Guatemala

Ingeniería en Ciencias y Sistemas

Lenguajes formales de programación

PROYECTO FINAL
MANUAL TECNICO: Análisis sintáctico

Cristian Giovanni Estrada Ramirez

Carnet 202006413

30 de abril del 2023

OBJETIVO:

El objetivo del software es proporcionar una herramienta de análisis y visualización de archivos de entrada. En primer lugar, el software analizará el archivo de entrada y verificará si cumple con el modelo estándar indicado en el proyecto. Se realizará una verificación de errores de sintaxis para asegurarse de que la estructura del archivo sea coherente y cumpla con los requisitos del modelo estándar. En caso de encontrar algún error, el software informará al usuario y proporcionará detalles sobre los errores detectados. Si no se detectan errores, el software procederá a procesar los datos del archivo de entrada.

Una vez que los datos del archivo de entrada se hayan verificado y validado, el software generará una visualización de las operaciones incluidas en el archivo. Esto puede ser en forma de gráficos, diagramas, tablas o cualquier otro formato visual que sea relevante para los datos procesados. La visualización permitirá al usuario entender mejor los datos incluidos en el archivo y analizarlos de manera más efectiva.

ESPECIFICACIONES TECNICAS:

Las especificaciones técnicas mediante las que fue elaborado el sistema son las siguientes:

- Hardware:
 - Procesador Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz 3.00 GHz
 - 8 GB de Ram
 - No se requirió el uso de ningún tipo de recursos gráficos extra
- Software:
 - Visual Studio Code
 - Sistema operativo: Windows 10 Pro
 - Sistema operativo de 64 bits

LOGICA DEL PROGRAMA:

Clases utilizadas:

- Analizador: Esta clase es la que realizará todo el análisis sintáctico
 - Variables
 - Líneas
 - Index
 - Fila
 - Columna
 - guardarFormula
 - ListaErrores
 - MaestroFormulas
 - TokenList
 - TokenID
 - Formula
 - TokenActual
 - DescErr
 - **_token():** La función verifica si el carácter actual en la línea actual no es un espacio en blanco. Si es así, llama a la función `_juntar` para unir los caracteres del token actual y luego llama a la función `_analizar` para verificar si el token es válido. Si el token es válido, actualiza el índice actual de la línea para que apunte al final del token y devuelve el estado siguiente. Si el token no es válido, devuelve el estado 'ERROR'. Si el carácter actual es un espacio en blanco, la función simplemente devuelve el estado actual sin hacer ninguna verificación adicional.
 - **_juntar():** toma dos argumentos: `"_index"` y `"_count"`. Su propósito es unir "n" líneas de texto comenzando desde la línea "index" y continuar hasta "index+n" líneas. Primero, se crea una cadena vacía llamada "tmp". Luego, se utiliza un bucle for para recorrer "n" líneas de texto desde la línea "index" hasta la línea "index+n". En cada iteración del bucle, se agrega el contenido de la línea actual a la cadena "tmp". Finalmente, se devuelve la cadena "tmp" que contiene la concatenación de todas las líneas de texto seleccionadas. Si ocurre alguna excepción durante la ejecución del bucle, la función devuelve "None".
 - **_analizar():** toma dos argumentos: `"token"` y `"texto"`. Su propósito es verificar si "texto" coincide con "token" en la misma posición de cada letra. Primero, se inicializa "count" a cero y se crea una variable temporal "tokem_tmp" vacía. A continuación, se realiza un bucle for a través de cada letra en "texto". Si la letra en la posición actual de "texto" coincide con la letra en la misma posición de "token", se agrega la letra a "tokem_tmp" y "count" se incrementa en uno. Si la letra no coincide, la función devuelve False.

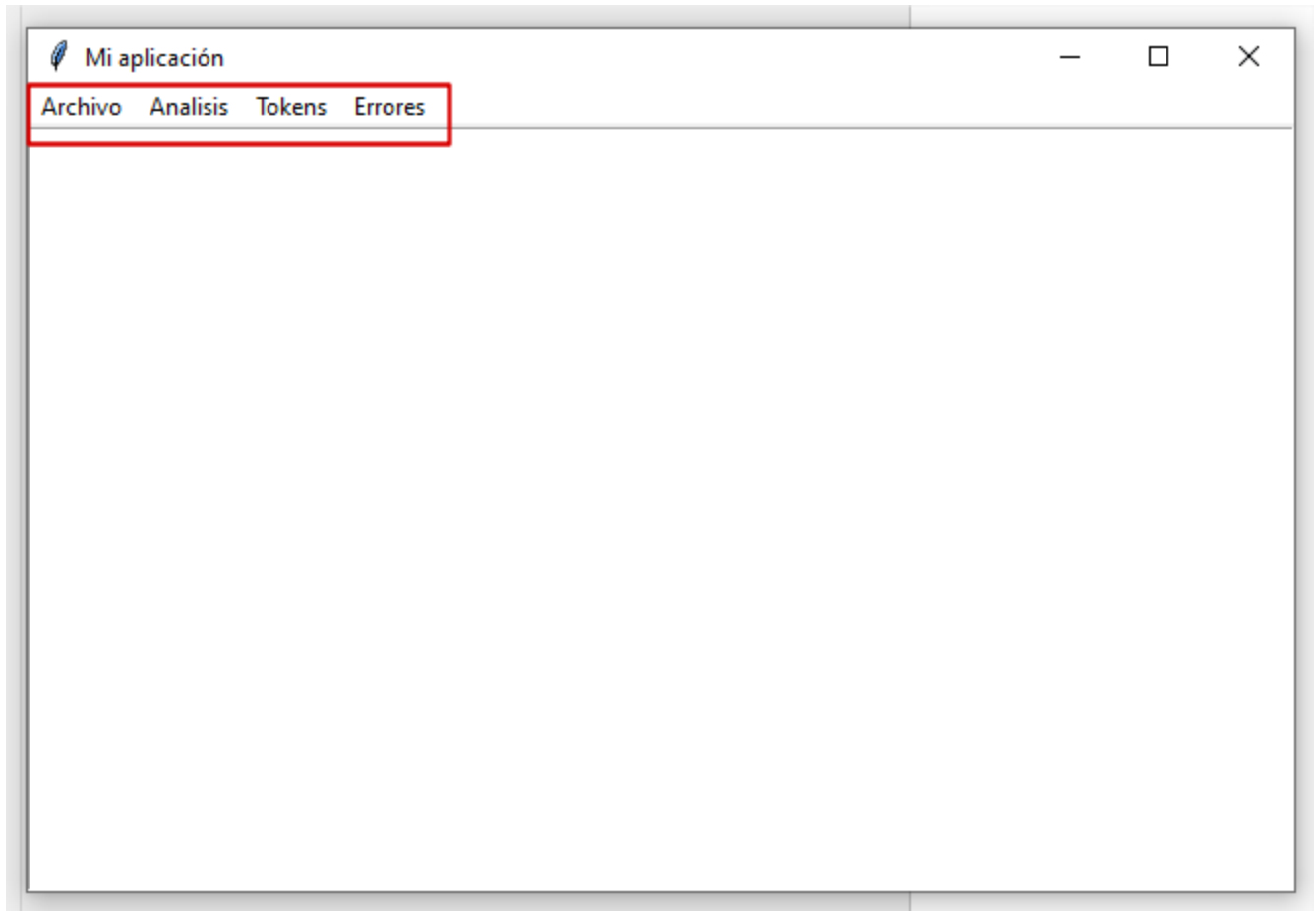
- **_analizarCadena():** La función `_analizarCadena` es una función auxiliar del analizador léxico que se utiliza para analizar una cadena de caracteres en una línea determinada. Comienza inicializando una variable llamada `estado_aux` en una cadena vacía y otra variable llamada `tmp` en el índice actual de la línea.

Luego, entra en un bucle `while` que se ejecuta mientras el carácter actual en la línea no es una cadena vacía. Si el carácter actual es una nueva línea, la función devuelve un estado de 'ERROR'. Si `estado_aux` está vacío, la función lo inicializa en 'INICIO'.

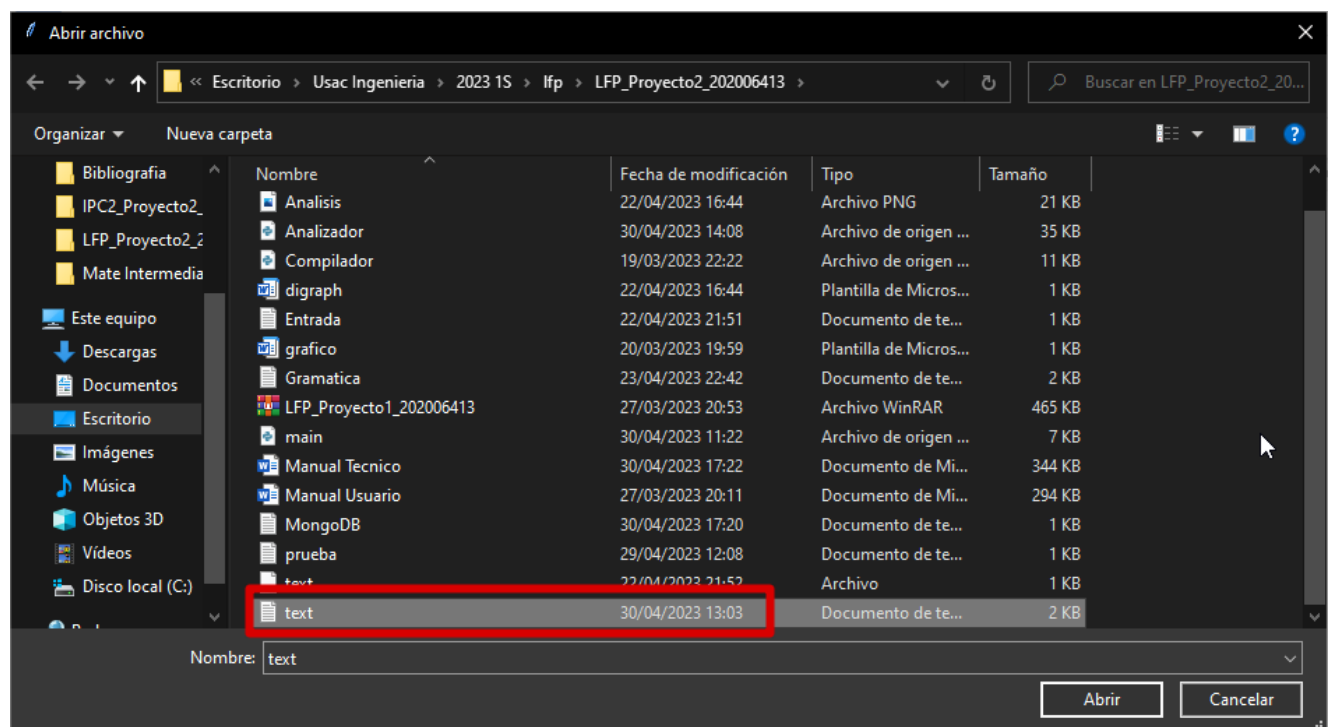
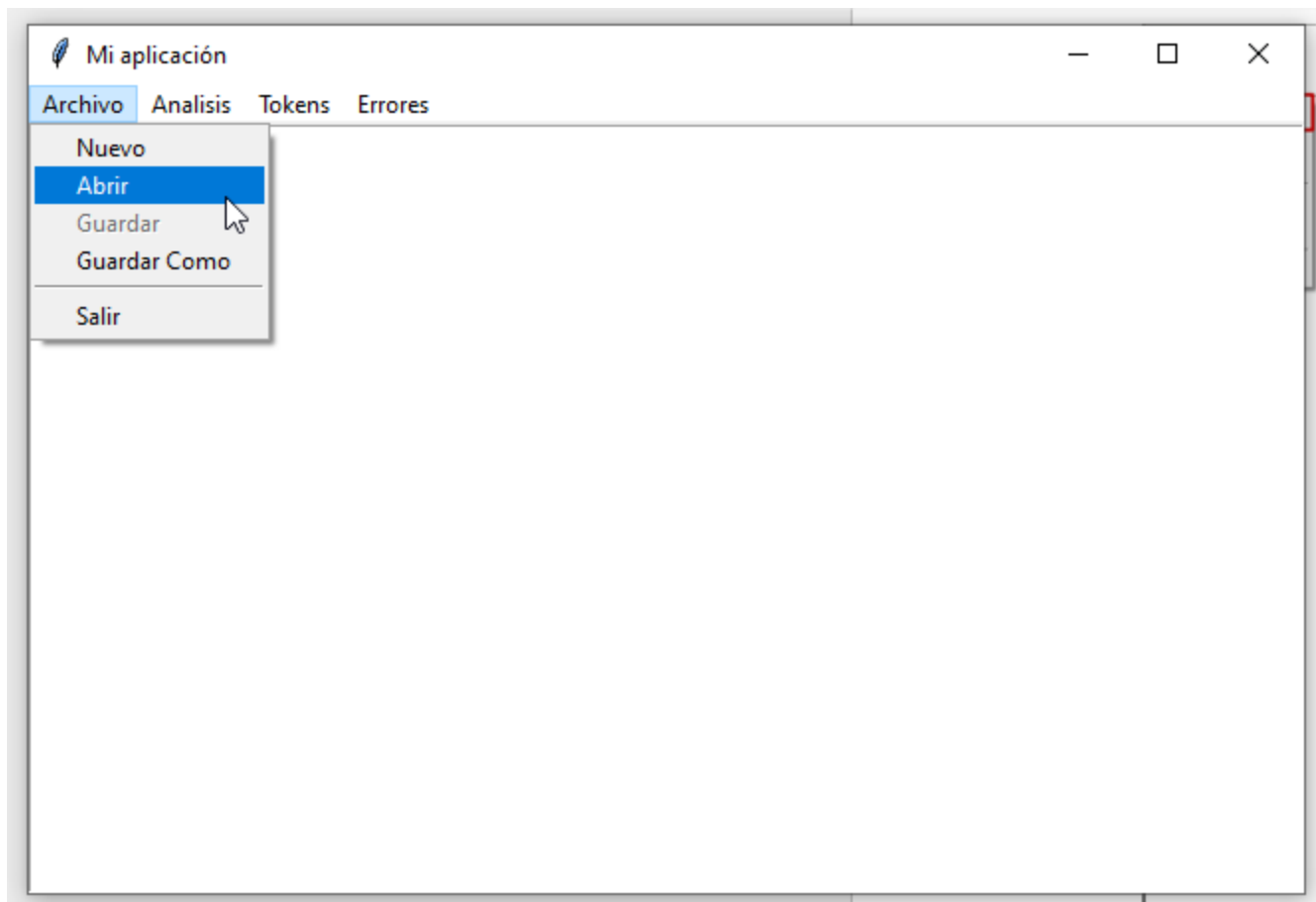
- **_analizarCadenaJSON ():** la función `_analizarCadenaJSON` se utiliza para acumular caracteres hasta que se encuentra un carácter especial o una cadena específica, momento en el que devuelve la cadena acumulada y la posición actual. Esta función es específica para el formato JSON y se utiliza en un analizador léxico.
- **_analizarJSON():** Esta función es parte de un analizador léxico de un archivo de entrada que contiene un objeto JSON. La función recibe el contenido del archivo de entrada como una lista de líneas y analiza cada carácter para identificar y clasificar cada token correspondiente a un objeto JSON.
- **_compile():** se encarga de analizar una serie de líneas de texto y crear una lista de tokens, donde cada token es una unidad léxica (palabra, símbolo, número, etc.) que representa una parte de la estructura de un lenguaje de programación.
La función tiene un ciclo `while` que se ejecuta mientras haya líneas de texto por analizar. Dentro del ciclo, se utilizan una serie de estados (`S0`, `S1`, `S2`, etc.) que representan las distintas fases del análisis léxico, y que permiten ir verificando cada unidad léxica en la línea actual.

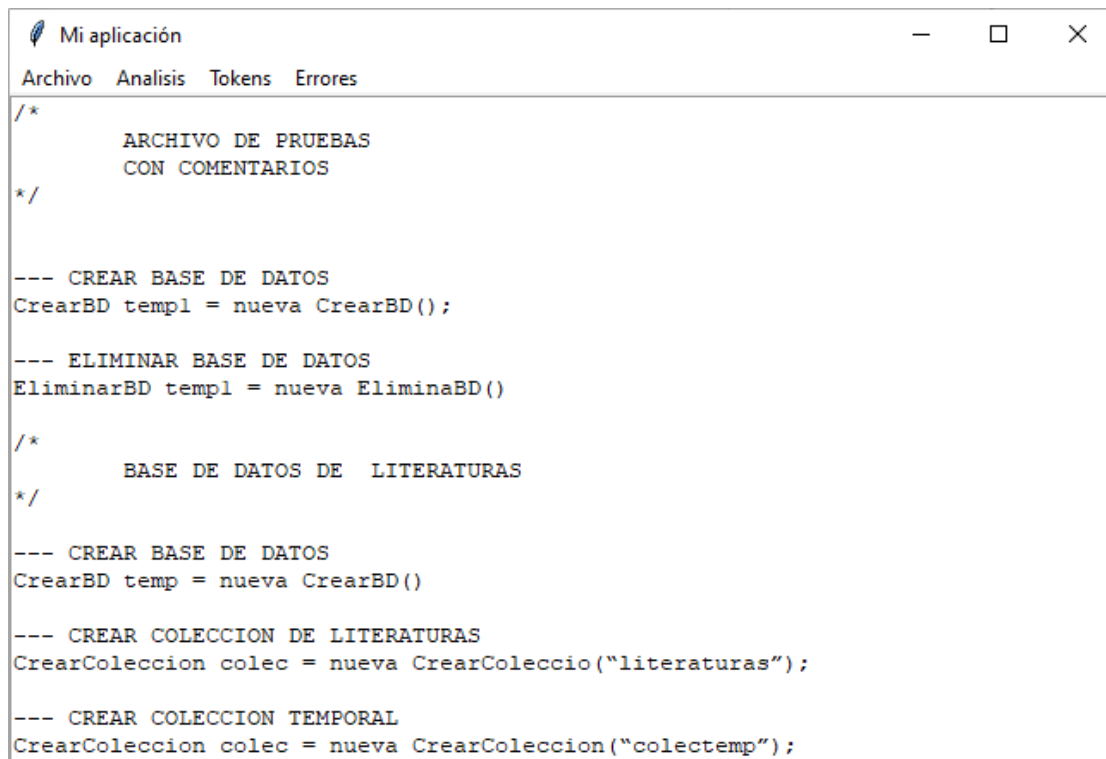
FLUJO DEL PROGRAMA:

Al iniciar el sistema abre una ventana en la cual se podrá editar texto, además tendremos las pestañas de Archivo, Análisis, Tokens y Errores con las cuales realizaremos todas las acciones:



Seleccionaremos las opciones que nos ofrece "Archivo", las cuales son Nuevo, Abrir, Guardar, Guardar Como y Salir, en este caso seleccionaremos la opción de abrir lo cual desplegará el explorador de archivos donde vamos a buscar el archivo de entrada:





```
/*
    ARCHIVO DE PRUEBAS
    CON COMENTARIOS
*/

--- CREAR BASE DE DATOS
CrearBD templ = nueva CrearBD();

--- ELIMINAR BASE DE DATOS
EliminarBD templ = nueva EliminaBD()

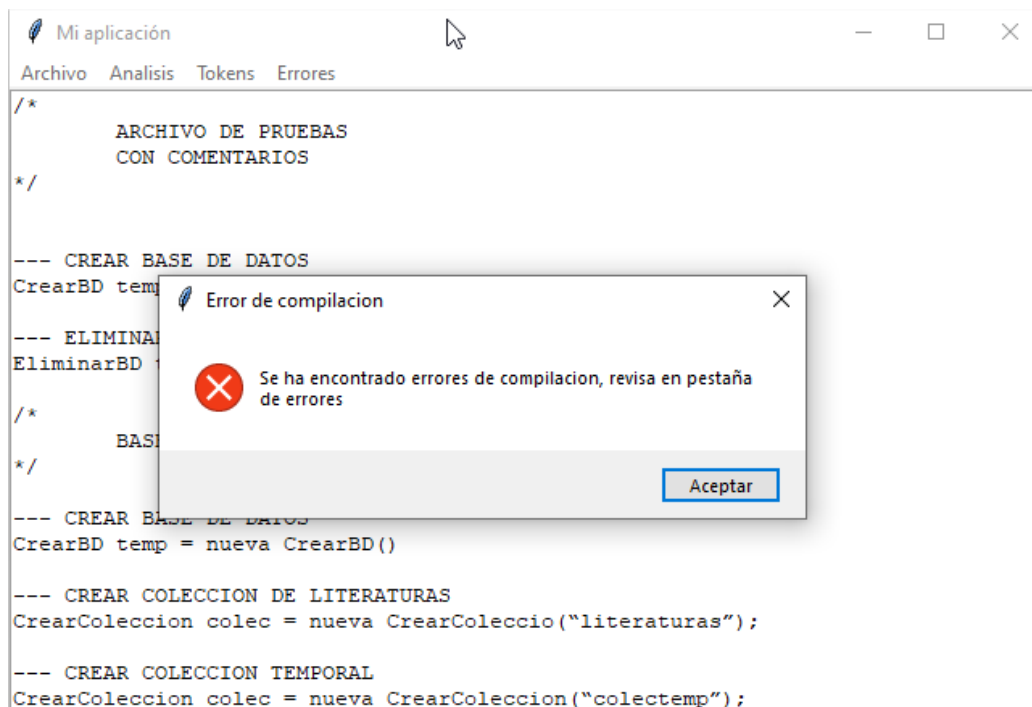
/*
    BASE DE DATOS DE LITERATURAS
*/

--- CREAR BASE DE DATOS
CrearBD temp = nueva CrearBD()

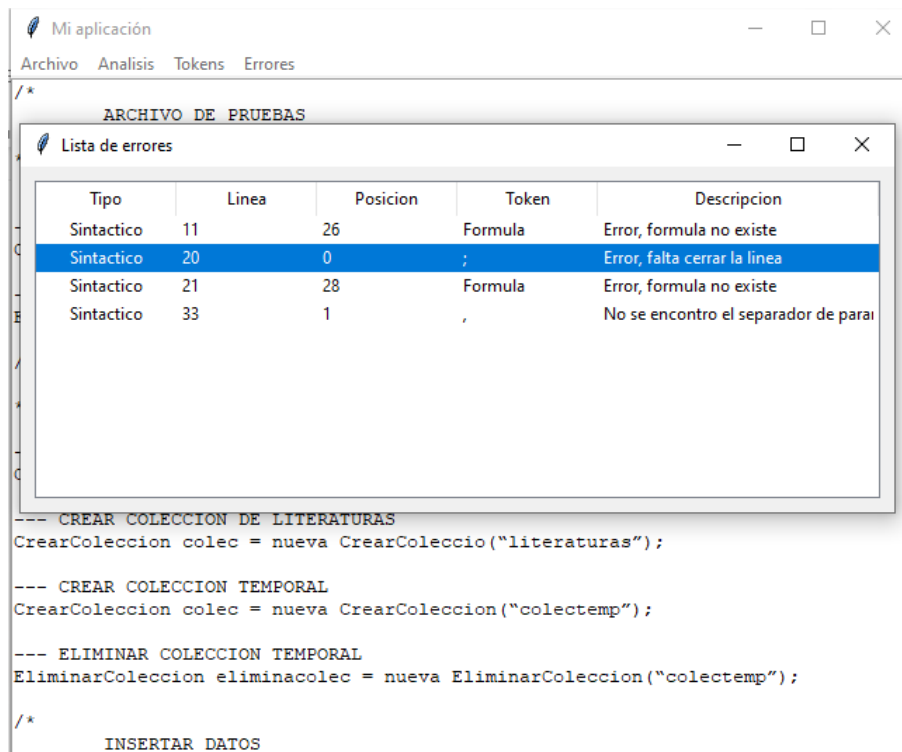
--- CREAR COLECCION DE LITERATURAS
CrearColeccion colec = nueva CrearColeccio("literaturas");

--- CREAR COLECCION TEMPORAL
CrearColeccion colec = nueva CrearColeccion("colectemp");
```

El archivo seleccionado se podrá en pantalla para que el usuario pueda hacer las modificaciones necesarias, en este caso no haremos ninguna modificación, lo que haremos es acceder a la pestaña de Análisis en la cual realizaremos un análisis sobre el código impreso en pantalla:



Como observamos el programa nos genera una alerta, la cual nos indica que nuestro código tiene errores los cuales deben ser solucionados para poder generar nuestras formulas de mongoBD, para poder ver los errores de compilación tendremos que ir a la pestaña de Errores y ver que nos indica:



Ahora realizaremos los cambios que nos están afectando la compilación, luego guardaremos los cambios y por ultimo volveremos a analizar:

```

--- CREAR BASE DE DATOS
CrearBD temp = nueva CrearBD();

--- CREAR COLECCION DE LITERATURAS
CrearColeccion colec = nueva CrearColeccion("literaturas");

--- CREAR COLECCION TEMPORAL
CrearColeccion colec = nueva CrearColeccion("colectemp");
  
```

```

--- ELIMINAR BASE DE DATOS
EliminarBD temp1 = nueva EliminarBD();

/*
    BASE DE DATOS DE LITERATURAS
*/

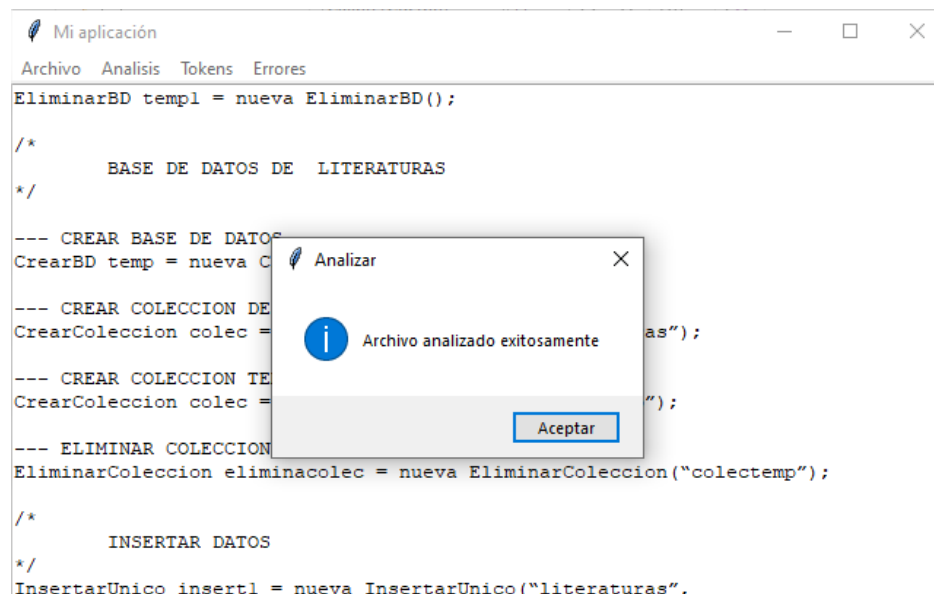
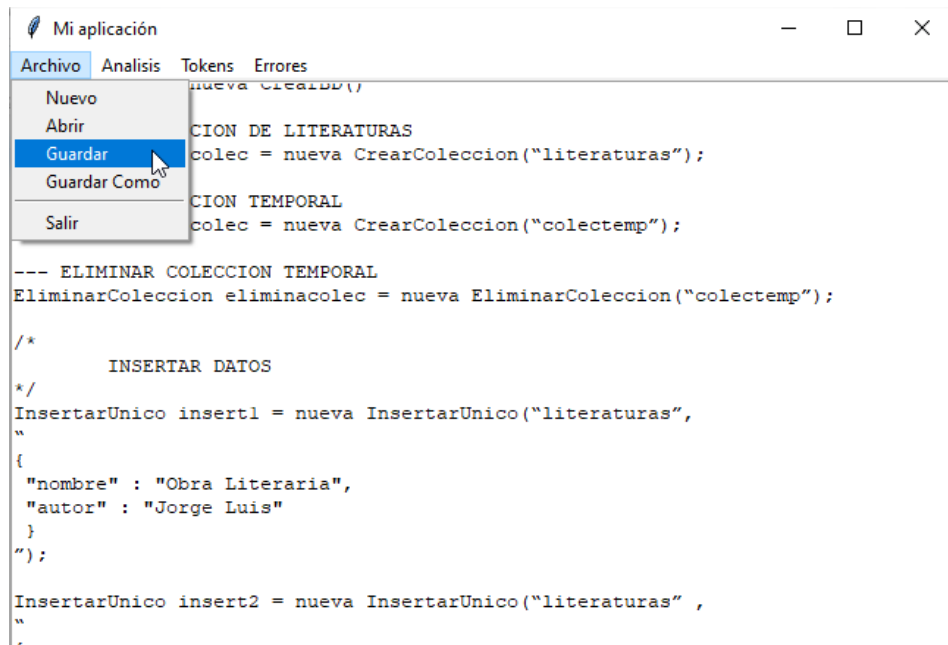
--- CREAR BASE DE DATOS
CrearBD temp = nueva CrearBD()

--- CREAR COLECCION DE LITERATURAS
CrearColeccion colec = nueva CrearColeccion("literaturas");

--- CREAR COLECCION TEMPORAL
  
```

```

/*
    INSERTAR DATOS
*/
InsertarUnico insert1 = nueva InsertarUnico("literaturas",
{
    "nombre" : "Obra Literaria",
    "autor" : "Jorge Luis"
});
  
```



Una vez analizado el programa generará un archivo el cual incluye todas las funciones insertadas en el archivo de entrada pero con el formato específico para el manejo de bases de datos de MongoDB:

```

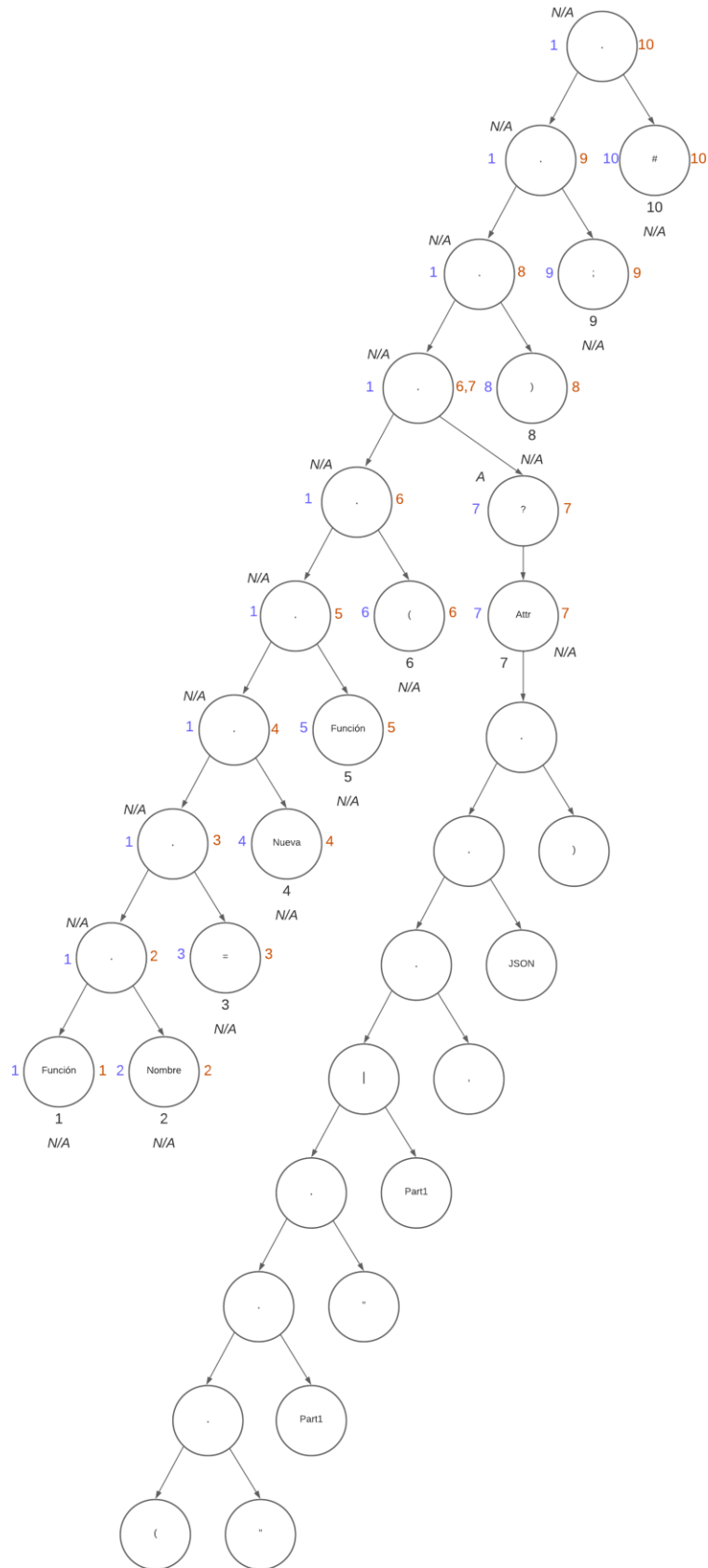
1  use('temp1');
2
3  temp1.dropDatabase();
4
5  use('temp');
6
7  dbo.createCollection(♦literaturas♦);
8
9  dbo.createCollection(♦colectemp♦);
10
11  dbo.colectemp.drop();
12
13  dbo.literaturas.InsertOne("
14  {
15      "nombre" : "Obra Literaria",
16      "autor" : "Jorge Luis"
17  }
18  ");
19
20  dbo.literaturas.InsertOne("
21  {
22      "nombre" : "El Principito",
23      "autor" : "Antoine de Saint"
24  }
25  ");
26
27  dbo.literaturas.InsertOne("
28  {
29      "nombre" : "Moldavita. Un Visitante Amigable",
30      "autor" : "Norma Mu♦oz Ledo"
31  }
32  ");
33
34  dbo.literaturas.InsertOne("
35  {
36      "nombre" : "Obra Literaria"
37  }
38  ");
39
40  dbo.literaturas.Find();
41
42  dbo.literaturas.InsertOne();
43

```

Por ultimo revisaremos que se hayan registrado los tokens en el programa, esto nos servirá para ver que pasos utilizó el programa para poder leer la estructura del archivo de entrada:

Anexos

Método del arbol:



autómata Finito determinista:

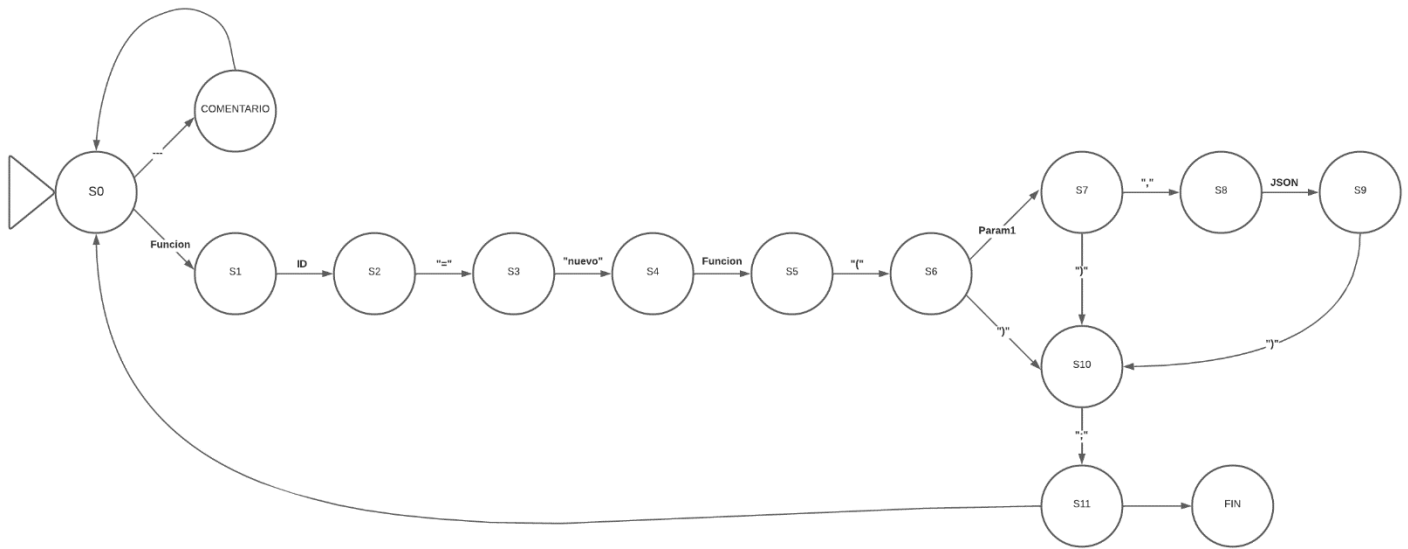


Tabla de tokens

Token	Definición	Uso
Funcion	Una funcion	Se utiliza para llamar a una funcion de mongoDB
ID	Es una variable	Es el nombre que se utiliza para hacer referencia a la declaracion
nueva	Palabra clave	Se utiliza para indicar que el objeto es nuevo
Param1	Cadena de texto	Es el nombre del carácter a ingresar
{ }	Apertura y cierre JSON	Se utiliza para encerrar datos en formato JSON
Val	Es una variable	Se utiliza para asignar el valor a un ID
=	Un operador	Se utiliza para asignar la expresion
;	Un delimitador	Se utiliza para indicar el fin de la instrucción