Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Laboratorio de Lenguajes Formales y de Programación

Manual Técnico

Proyecto 2: Analizador léxico y Sintáctico

Catedrático: Mario Josué Solis Solórzano

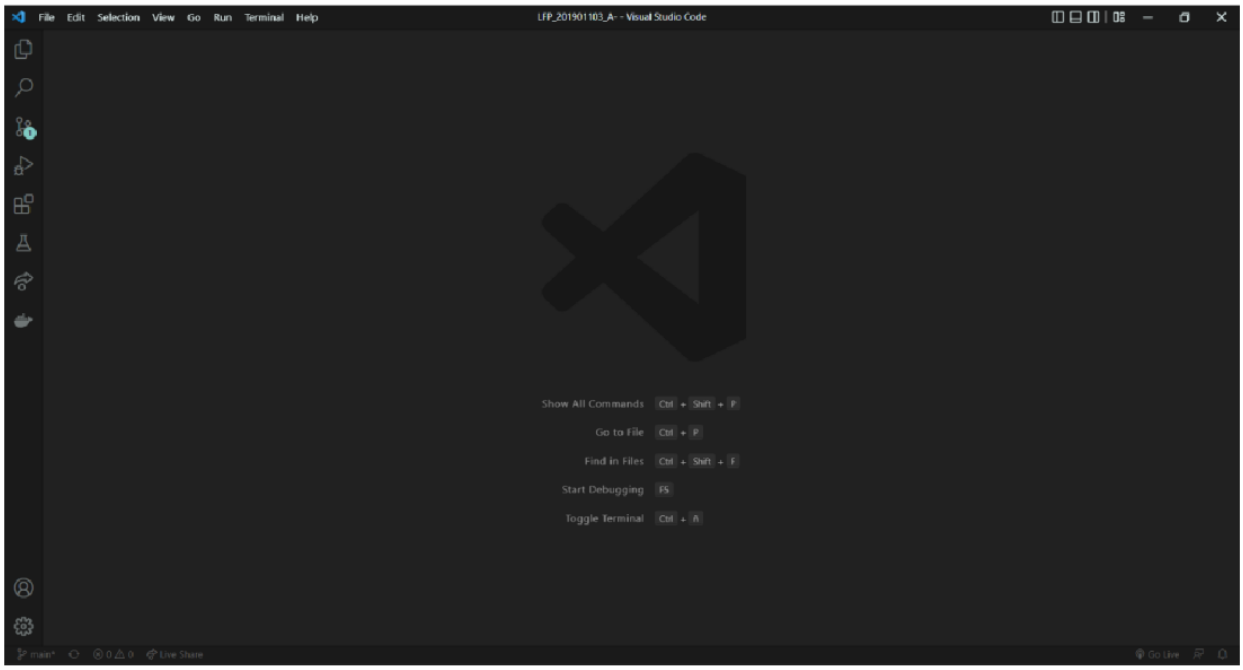
Sección: A-

Nombre: Josué Daniel Rojché García

Carné: 201901103

Requerimientos del Sistema

En la realización del software el Editor de texto que se utilizó fue Visual Studio Code.



El programa fue realizado con la versión de Python 3.11.1 para sistemas operativos de 64 bits.

Analizador Léxico

Para que el programa funcione correctamente se realizaron diversas validaciones y manejo de errores para que no finalice indebidamente durante la ejecución de esta. Por lo cual explicaremos lo que realiza el código.

Primero se importan las librerías necesarias, las cuales se utilizarán más adelante, por ejemplo: la librería tkinter, la cual servirá para las ventanas graficas

```
import os
import tkinter as tk
from tkinter import CENTER, INSERT, RIGHT, Y, Scrollbar, StringVar, filedialog, Tk, ttk
from tkinter.messagebox import showerror, showinfo, showwarning ,askquestion
import webbrowser
from automataFD import AFD
from argparse import ONE_OR_MORE
```

Menú Principal

La pantalla se crea con tk.Tk(), con title se le agrega un titulo a la pestaña, con geometry indicamos el tamaño que tendrá dicha ventana indicando el ancho por altura, con configure se indica el color de fondo que tendrá la ventana, resizable sirve para indicar que la ventana sea de tamaño fijo y que el usuario no pueda agrandarla o hacerla más pequeña.

Para crear una barra de menú se utiliza tk.Menu(), el cual sirve para agregarle los datos que sean necesarios, en este caso se crea el espacio para el manejo de los archivos por lo que se utiliza add_cascade() con sus respectivos parámetros, el primer parámetro indica que se agregará al menú de barra, y el label le indica el nombre que tendrá para esas opciones, para agregar las subopciones se utiliza add_command() el cual lleva como parámetros los siguientes, label el cual indica el nombre de la opción a utilizar, luego command al cual se le asigna el método al que accederá para realizar la acción que corresponda, luego con state se indica que la opción no estará disponible si no se carga un archivo con anterioridad.

Con tk.Text() se agrega dos áreas de texto que servirán para editar el archivo y mostrar los errores del archivo al momento de realizar las opciones que correspondan a partir que se cargue el archivo en el sistema, con configure() se le indica el color de fondo que tendrá con bg, y state para indicar que al principio no se pueda editar en el área de texto.

Con tk.label() se crea un texto en una línea que estará plasmado para indicar cual cuadro de texto es para el archivo cargado y el que muestra los errores.

Con config() se agrega el parámetro que agrega a la ventana la barra de menú y con mainloop() se indica que que la ventana será visible para el usuario.

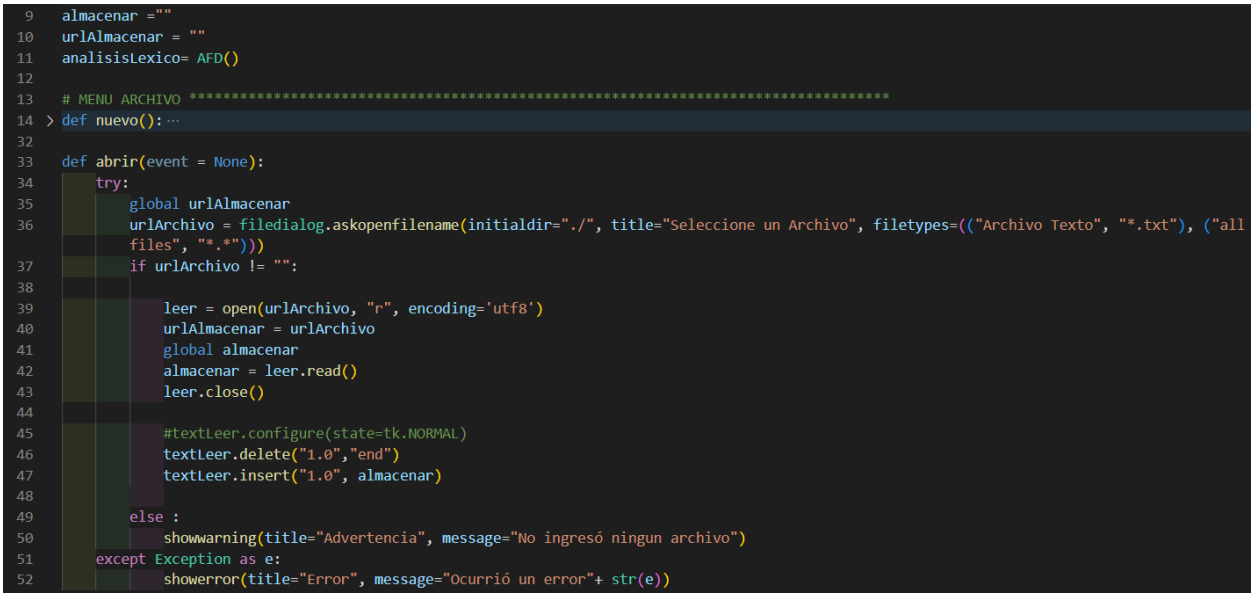
```
try:
    menu = tk.Tk()
    menu.title("PROYECTO NO.2")
    menu.geometry("800x650")
    menu.configure(bg="#212F3C")
    menu.resizable(False, False)
    #Crea la barra del menu
    barra_Menu = tk.Menu()
    #Crea el primer elemento el cual será para archivo y se enlaza ala barra
    menuArchivo = tk.Menu(barra_Menu, tearoff=False)
    #Al add_command se le puede pasar el parametro accelerator="Ctr+N" para agregar un atajo con el teclado
    menuArchivo.add_command(label="Abrir",accelerator="Ctrl+N",command= abrir)
    menu.bind_all("<Control-n>",abrir )
    menuArchivo.add_command(label="Nuevo", command= nuevo)
    menuArchivo.add_command(label="Guardar", command= guardar)
    menuArchivo.add_command(label="Guardar Como",command=guardarComo)
    menuArchivo.add_separator()
    menuArchivo.add_command(label="Inicializar", command=inicializar)
    menuArchivo.add_separator()
    menuArchivo.add_command(label="Salir", command=menu.quit,activebackground="Red")
    menuAnalisis= tk.Menu(barra_Menu, tearoff=False)
    menuAnalisis.add_command(label="Generar Sentencias MongoDB",command= generarSentenciasMDB)
    menuTokens = tk.Menu(barra_Menu, tearoff= False)
    menuTokens.add_command(label="Ver Tokens", command= verTokens)
    menuErrores = tk.Menu(barra_Menu, tearoff= False)
    menuErrores.add_command(label="Ver Errores", command= verErrores)
    # a la barra menú le agregamos el menuArchivo
    barra_Menu.add_cascade(menu= menuArchivo, label= "Archivo")
    barra_Menu.add_cascade(menu= menuAnalisis, label= "Análisis")
    barra_Menu.add_cascade(menu= menuTokens, label= "Tokens")
    barra_Menu.add_cascade(menu= menuErrores, label= "Errores")
    posX = StringVar()
    posY = StringVar()
    def posicionXY(event):
        global posX
        global posY
        xy = textLeer.index(INSERT)
```

```
        xy = textLeer.index(INSERT)
        aux = xy.split('.')
        posX.set(aux[0])
        posY.set(aux[1])
        #print('Coordenada x:',posX, 'Coordenada y', posY)

    textLeer = tk.Text()
    textLeer.configure(bg="#C8C885")
    textLeer.place(x= 5, y =5, height= 600, width= 690)
    textLeer.insert("1.0", "---Area de edición de codigo.")
    textLeer.bind("<Button-1>", posicionXY)
    menu.config(menu=barra_Menu)
    anchoTotal = menu.winfo_screenwidth()
    altoTotal = menu.winfo_screenheight()
    posicionAncho = round(anchoTotal/2 - 700/2)
    posicionAlto = round(altoTotal/2 - 650/2)

    menu.geometry("800x650"+"+"+str(posicionAncho)+"+"+str(posicionAlto))

    label1 = tk.Label(menu, text="Archivo abierto", bg="#212F3C",fg="#FFFFFF",width= 20, font=("Arial", 13)).place(x= 250, y =615)
    tk.Label(menu, text="Linea:", bg="#212F3C", fg="#FFFFFF", font=("Arial", 10)).place(x= 710, y =10)
    tk.Label(menu, text="Columna:", bg="#212F3C", fg="#FFFFFF", font=("Arial", 10)).place(x= 710, y =80)
    tk.Label(menu, textvariable = posX, bg="#212F3C", fg="#FFFFFF",font=("Arial", 10)).place(x =750,y=40)
    tk.Label(menu, textvariable = posY, bg="#212F3C", fg="#FFFFFF",font=("Arial", 10)).place(x =750,y=100)


    menu.mainloop()
except Exception as e:
        showerror(title="Error", message="Ocurrió un error"+str(e))
```

Método abrir

Primero se crea la variable almacenar la cual servirá para guardar todo el texto que tiene el archivo a abrir, la variable url guardará la dirección donde se encuentra dicho archivo, y para manejar el analizador realizamos la instancia hacia la clase AFD().

Ahora en el método abrir se utiliza filedialog.askopenfilename() para abrir una ventana que permite acceder a los archivos y seleccionar el que se requiera utilizar en el programa, el cual debe ser con la extensión .txt, luego con if se valida si se ha seleccionado algún archivo, si el archivo es cargado entonces abre el archivo con el método open, y con read() lee el contenido del mismo y lo almacena en la variable creada globalmente, además se agregan estos datos al cuadro de texto que se creó inicialmente para poder editar el contenido del mismo, y se habilitan las demás opciones de la barra de menú con entryconfig, y con showarning indicamos un mensaje para dar a entender al usuario que no cargó ningún archivo en el sistema.



```
9   almacenar =""
10  urlAlmacenar = ""
11  analisisLexico= AFD()
12
13  # MENU ARCHIVO ********************************************************************************
14  def nuevo():···
32
33  def abrir(event = None):
34      try:
35          global urlAlmacenar
36          urlArchivo = filedialog.askopenfilename(initialdir="./", title="Seleccione un Archivo", filetypes=(("Archivo Texto", "*.txt"), ("all
            files", "*.*")))
37          if urlArchivo != "":
38
39              leer = open(urlArchivo, "r", encoding='utf8')
40              urlAlmacenar = urlArchivo
41              global almacenar
42              almacenar = leer.read()
43              leer.close()
44
45              #textLeer.configure(state=tk.NORMAL)
46              textLeer.delete("1.0","end")
47              textLeer.insert("1.0", almacenar)
48
49          else :
50              showwarning(title="Advertencia", message="No ingresó ningun archivo")
51      except Exception as e:
52              showerror(title="Error", message="Ocurrió un error"+ str(e))
```

Método guardar

Se crea una variable que tendrá el método open para acceder al archivo que se abrió y poder editar en él, guardando lo que obtenga del área de texto con el método get, y este se escribe con write en el archivo, luego se cierra esté para evitar el seguir usándolo sin ser necesario.

```python
def guardar():
    try:
        global almacenar
        saveArchivo = open(urlAlmacenar, "w", encoding='utf8')
        saveArchivo.write(textLeer.get("1.0","end"))
        saveArchivo.close()

        almacenar = str(textLeer.get("1.0","end"))
        analisisLexico.limpiarDatos()
    except Exception as e:
            showerror(title="Error", message="Ocurrió un error")
```

Método guardar como

Se crea el método para permitir guardar lo que se editó en el cuadro de texto con otro nombre y en la ubicación que el usuario requiera, para ello se comienza creando la variable con filedialog.asksaveasfilename() el cual permite abrir la ventana para seleccionar la ruta donde se guardará el archivo y agregarle en nombre al mismo.

Se valida con if si el usuario ha ingresado alguna ruta y agregado un nombre al archivo, si el usuario ingresa correctamente los datos se utiliza open para crear el archivo y permitir escribir en el con el método write() y este obtiene con get los datos que se encuentran en el cuadro de texto y para finalizar se cierra el archivo con close().

Si el usuario cancela la opción de guardar como, entonces se muestra un mensaje de advertencia indicando que puede perder los datos si no los guarda.

```python
def guardarComo():
    try:
        guardar_Como = filedialog.asksaveasfilename(initialdir="./", title="Guardar Como", filetypes=(("Archivo texto", ".txt"), ("all
        files", "*.*")))
        if guardar_Como != "":
            saveComoArchivo = open(guardar_Como +".txt", "w", encoding='utf8')
            saveComoArchivo.write(textLeer.get("1.0","end"))
            saveComoArchivo.close()

            showinfo(title="Guardado", message="Archivo guardado exitosamente")
        else :
            showwarning(title="Advertencia", message="¡Si no guarda el archivo se perderan los datos!")
    except Exception as e:
            showerror(title="Error", message="Ocurrió un error")
```

Método nuevo

Se crea el método para permitir crear un nuevo archivo, por lo que se accede a la variable que almacena la url, en el caso de que se haya abierto un archivo anteriormente, con un if se valida si esta variable se encuentra vacía, ya que si está vacía simplemente se borrarán los datos con el método inicializar. Si de lo contrario, la variable no está vacía, entonces se abre un mensaje en pantalla con el método askquestion() indicando que si no guarda los datos serán borrados, si el usuario presiona “No” entonces se abrirá una ventana para guardar el archivo de lo contrario, simplemente se llama al método inicializar().

```
14  def nuevo():
15      global urlAlmacenar
16      try:
17          if urlAlmacenar == "":
18              inicializar()
19              #textLeer.delete("1.0","end")
20              #textLeer.insert("1.0", "---Area de edición de codigo.")
21          else:
22              res = askquestion(title="Advertencia", message="¿Está seguro que desea salir sin guardar?\nSi desea guardar presione No, de lo
                contrario presione Sí.")
23
24              if res != "yes":
25                  guardarComo()
26              else:
27                  inicializar()
28                  #textLeer.delete("1.0","end")
29                  #textLeer.insert("1.0", "---Area de edición de codigo.")
30      except Exception as e:
31              showerror(title="Error", message="Ocurrió un error")
```

Método generarSentenciasMDB

Si la variable contiene datos, entonces accede al método analizando() de la clase AFD(), y se envía como parámetro a la variable con los datos, con el método analizadorSintactico() se realizan las validaciones correspondientes para verificar que la estructura de las funciones sea correctas. Luego se crean dos variables booleanas que almacenaran el resultado de evaluar si las listas de errores léxicos o sintácticos está vacía y así poder escribir el archivo con el método escribiendoArchivo(), el cual obtendrá la cadena escrita con las funciones en mongoDB y así poder guardar esos datos con el método open y write, y de lo contrario mostrar un mensaje indicando que existen errores que deben ser corregidos.

```
81  def generarSentenciasMDB():
82      try:
83          analisisLexico.analizando(almacenar)
84          analisisLexico.analizadorSintactico()
85          #analisisLexico.imprimir_tokensSintacticos()
86          #analisisLexico.imprimir_ErroresSintacticos()
87
88          # Si es True es porque la lista está vacía
89          booleanoLexico = analisisLexico.vacioTablaErrorLexico()
90          booleanoSintactico = analisisLexico.vacioTablaErrorSintactico()
91          if ((booleanoLexico == True) and (booleanoSintactico == True)):
92              textoEscribir = analisisLexico.escribiendoArchivo()
93
94              saveArchivo = open(urlAlmacenar, "w", encoding='utf8')
95              saveArchivo.write(textoEscribir)
96              saveArchivo.close()
97
98
99          else:
100             showwarning(title="Advertencia", message="Por favor corregir los errores.\nVer tabla de errores")
101         #validar si existen errores en la tabla de errores de token y de sintactico
102         #Si no existen errores generar las sentencias mongoDB
103     except Exception as e:
104             showerror(title="Error", message="Ocurrió un error")
```

Método verTokens

Este método tiene la finalidad de abrir una ventana en la cual se mostraran los datos de la tabla de tokens, por lo cual se crea la ventana con Tk.Toplevel para indicar que es secundaria, luego se proporcionan titulo, tamaño, color y se agrega una tabla con Treeview y se proporciona un estilo con style, y una barra para poder visualizar mas datos si la tabla crece aún más, además se agregan las columnas con colum, y los encabezados de los mismos con heading, para luego agregar con tag_configure un color predeterminado a las filas.

```
107  def verTokens():
108      try:
109          auxiliarTablaTokens = analisisLexico.obtenerTablaTokens()
110
111          ventana_Token = tk.Toplevel()
112          ventana_Token.title("Tabla TOKENS")
113          ventana_Token.geometry("650x600")
114          ventana_Token.configure(bg="yellow")
115          ventana_Token.resizable(False, False)
116
117          style = ttk.Style()
118          style.theme_use("default")
119          style.configure("Treeview", background="silver", foreground="black", rowheight=30, fieldbackground="silver")
120          style.map("Treeview", background=[("selected", "green")])
121
122          scroll_bar = Scrollbar(ventana_Token)
123          scroll_bar.pack(side=RIGHT, fill=Y)
124
125          tablaDinamica = ttk.Treeview(ventana_Token, yscrollcommand=scroll_bar.set, columns=("col1", "col2"),height= 500)
126          scroll_bar.config(command=tablaDinamica.yview)
127          tablaDinamica.column("#0", width=80)
128          tablaDinamica.column("col1", width=200, anchor=CENTER)
129          tablaDinamica.column("col2", width=300, anchor=CENTER)
130
131          tablaDinamica.heading("#0", text="Correlativo", anchor=CENTER)
132          tablaDinamica.heading("col1", text="Token", anchor=CENTER)
133          tablaDinamica.heading("col2", text="Lexema", anchor=CENTER)
134      # agregando estilo a las filas
135          tablaDinamica.tag_configure("oddrow", background="white")
136          tablaDinamica.tag_configure("evenrow", background="lightblue")
137      # AGREGANDO LISTA DE OBJETOS A LA TABLA DE ACUERDO AL TAMAÑO DE LA LISTA.
138          iterador = 1
```

Con for se recorre la tabla de tokens y se agregan los datos a la tabla con el método insert, además la tabla se agrega a la pantalla con pack y con mainloop es visible.

```
140          for j in auxiliarTablaTokens:
141              t_t = j.tok
142              t_lexema = j.lexema
143
144              # MEJOR SE VA A MANEJAR CON WHILE PARA RECORRER LA LISTA OBJETOS.
145              if iterador % 2 == 0:
146                  tablaDinamica.insert("", tk.END, text=str(iterador), values=(t_t,t_lexema), tags=("evenrow",))
147              else:
148                  tablaDinamica.insert("", tk.END, text=str(iterador), values=(t_t,t_lexema), tags=("oddrow",))
149
150              iterador += 1
151          tablaDinamica.pack(pady=20)
152
153          ventana_Token.mainloop()
154
155      except Exception as e:
156              showerror(title="Error", message="Ocurrió un error"+ str(e))
```

Método verErrores

Este método tiene la finalidad de abrir una ventana en la cual se mostraran los datos de la tabla de errores léxicos y sintacticos, por lo cual se crea la ventana con Tk.Toplevel para indicar que es secundaria, luego se proporcionan titulo, tamaño, color y se agrega una tabla con Treeview y se proporciona un estilo con style, y una barra para poder visualizar más datos si la tabla crece aún más, además se agregan las columnas con colum, y los encabezados de los mismos con heading, para luego agregar con tag_configure un color predeterminado a las filas.

```
def verErrores():
    try:
        auxiliarTablaErrores = analisisLexico.obtenerTablaErrores()
        auxiliarTErroresSintacticos = analisisLexico.obtenerTablaSintacticoErrores()
        ventana_Errores = tk.Toplevel()
        ventana_Errores.title("Tabla Errores")
        ventana_Errores.geometry("900x600")
        ventana_Errores.configure(bg="yellow")
        ventana_Errores.resizable(False, False)

        style = ttk.Style()
        style.theme_use("default")
        style.configure("Treeview", background="silver", foreground="black", rowheight=30, fieldbackground="silver")
        style.map("Treeview", background=[("selected", "green")])

        scroll_bar = Scrollbar(ventana_Errores)
        scroll_bar.pack(side=RIGHT, fill=Y)

        tablaDinamica = ttk.Treeview(ventana_Errores, yscrollcommand=scroll_bar.set, columns=("col1", "col2", "col3", "col4"),height= 500)
        scroll_bar.config(command=tablaDinamica.yview)
        tablaDinamica.column("#0", width=80)
        tablaDinamica.column("col1", width=80, anchor=CENTER)
        tablaDinamica.column("col2", width=80, anchor=CENTER)
        tablaDinamica.column("col3", width=300, anchor=CENTER)
        tablaDinamica.column("col4", width=300, anchor=CENTER)

        tablaDinamica.heading("#0", text="Tipo de Error", anchor=CENTER)
        tablaDinamica.heading("col1", text="Fila", anchor=CENTER)
        tablaDinamica.heading("col2", text="Columna", anchor=CENTER)
        tablaDinamica.heading("col3", text="Lexema o Token", anchor=CENTER)
        tablaDinamica.heading("col4", text="Descripción", anchor=CENTER)
    # agregando estilo a las filas
        tablaDinamica.tag_configure("oddrow", background="white")
        tablaDinamica.tag_configure("evenrow", background="lightblue")
    # AGREGANDO LISTA DE OBJETOS A LA TABLA DE ACUERDO AL TAMAÑO DE LA LISTA.
        iterador = 1
```

Con for se recorre la tabla de errores, tanto léxicos como sintacticos y se agregan los datos a la tabla con el método insert, además la tabla se agrega a la pantalla con pack y con mainloop es visible la ventana.

```
        for j in auxiliarTablaErrores:
            t_fila = j.fila
            t_columna = j.columna
            t_lexema = j.lexema
            t_des = j.tok

            # MEJOR SE VA A MANEJAR CON WHILE PARA RECORRER LA LISTA OBJETOS.
            if iterador % 2 == 0:
                tablaDinamica.insert("", tk.END, text='Lexico', values=(t_fila, t_columna, t_lexema, t_des), tags=("evenrow",))
            else:
                tablaDinamica.insert("", tk.END, text='Lexico', values=(t_fila, t_columna, t_lexema, t_des), tags=("oddrow",))

            iterador += 1

        #Agregar la lectura de errores sintacticos con otro for
        iterador1 = 1
```

```
    #Agregar la lectura de errores sintacticos con otro for
    iterador1 = 1

    for k in auxiliarTErroresSintacticos:
        t_fila1 = k.fila
        t_columna1 = k.columna
        t_lexema1 = k.lexema
        t_des1 = k.tok

        # MEJOR SE VA A MANEJAR CON WHILE PARA RECORRER LA LISTA OBJETOS.
        if iterador1 % 2 == 0:
            tablaDinamica.insert("", tk.END, text='Sintactico', values=(t_fila1, t_columna1, t_lexema1, t_des1), tags=("evenrow",))
        else:
            tablaDinamica.insert("", tk.END, text='Sintactico', values=(t_fila1, t_columna1, t_lexema1, t_des1), tags=("oddrow",))

        iterador1 += 1

    tablaDinamica.pack(pady=20)

    ventana_Errores.mainloop()
except Exception as e:
        showerror(title="Error", message="Ocurrió un error")
```

Método inicializar

Este método tiene la finalidad de limpiar los datos del programa y poder comenzar a utilizarlo otra vez sin la necesidad de cerrarlo y volverlo a abrir, por lo que se accede a todas las variables del sistema y se limpian, también se accede al método limpiarDatos para borrar los datos del analizador léxico, y con delete se borran los datos del área de edición de texto.

```
def inicializar():
    try:
        global almacenar
        almacenar = ""
        global urlAlmacenar
        urlAlmacenar = ""
        analisisLexico.limpiarDatos()
        textLeer.delete("1.0","end")
        textLeer.insert("1.0", "---Area de edición de codigo.")
        #textLeer.configure(state="disabled")
    except Exception as e:
            showerror(title="Error", message="Ocurrió un error")
```

Clase Token

En esta clase se ingresan fila, columna y lexema en el constructor, los cuales servirán para agregar en conjunto los datos a la tabla de tokens.

```
class Token:
    def __init__(self, fila, columna, lexema, tok):
        self.fila = fila
        self.columna = columna
        self.lexema = lexema
        self.tok = tok
```

Clase AFD

En el método constructor se inicializan las listas que continene las letras para comentarios, identificación, nueva, json, función, y tipo de funcion que podrá leer el autómata, se inicializa las variables necesarias a utilizar, como fila, columna, los estados actual, anterior, estados finales, así como las listas para almacenar los tokens léxicos y sintácticos, y las listas para almacenar los errores de ambos, etc.

```python
from token import Token

class AFD:
    def __init__(self):
        self.letrasComentarios = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q','r', 's', 't',
        'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-','{','}','[',']','.']
        self.identificacion = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q','r', 's', 't', 'u',
        'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9','_']
        self.nueva = ['n','u','e','v']
        self.letrasJson = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q','r', 's', 't', 'u', 'v',
        'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-', '{', '}', '"', '.', ',', '$', ':'] #'\n',' ', '\t'
        self.tipoFuncion = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q','r', 's', 't', 'u',
        'v', 'w', 'x', 'y', 'z']
        self.reservadasFunciones =['CrearBD','EliminarBD','CrearColeccion','EliminarColeccion','InsertarUnico','ActualizarUnico',
        'EliminarUnico','BuscarTodo','BuscarUnico']
        self.fila = 1
        self.columna = 0
        self.estadoActual = 'A'
        self.estadoAnterior = ''
        self.estadoFinal = ['L', 'J']
        self.tabla = []
        self.tablaErrores = []
        self.tablaSintactico = []
        self.tablaErroresSintacticos = []
        self.sentencias = ''
```

Método analizando

Este método recibe como parámetro el texto que se obtiene del archivo a analizar, el texto se almacena en una variable para manejar los datos localmente, por lo que se utiliza un bucle while para recorrer todos los datos del texto, luego se accede a cada carácter del texto, para luego comenzar con las validaciones de if que indicaran en que estado se encuentra y que carácter puede almacenar en base a el diseño del Autómata Finito Determinista, el cual encuentra el diseño al final de este documento.

Básicamente lo que se realiza es repetitivo, solamente se valida que se encuentre en un estado en especifico y luego cuando ingrese al carácter que coincida, envía el carácter como parámetro al método almacenarToken(), el cual sirve para enviar los datos a la lista de tokens. Luego se indica el estado actual como estado anterior, y el estado actual es el siguiente estado del diseño del AFD, y con else validamos que si el carácter no es admitido por el autómata, entonces se trata de un error, por lo cual es necesario almacenar el carácter en el método almacenarError().

```
22      def analizando(self, texto1):
23          tok = ''
24          # Eliminando espacios y saltos de linea de la cadena
25          validandoError = False
26          texto = texto1
27          # recorriendo el texto
28          while len(texto) > 0:
29              caracter = texto[0]
30              # validaciones de acuerdo al caracter que se está leyendo
31              if self.estadoActual == 'A':
32                  if caracter.lower() in self.tipoFuncion:
33                      tok += caracter
34                      self.estadoAnterior = 'A'
35                      self.estadoActual = 'B'
36                  elif caracter == '/':
37                      self.almacenarToken(caracter,"barra")
38                      self.estadoAnterior = 'A'
39                      self.estadoActual = 'C'
40                  elif caracter == '-':
41                      self.almacenarToken(caracter,"guion")
42                      self.estadoAnterior = 'A'
43                      self.estadoActual = 'D'
44                  elif caracter == '\n':
45                      self.fila += 1
46                      self.columna = 0
47                      texto = texto[1:]
48                      self.estadoAnterior = 'A'
49                      self.estadoActual = 'A'
50                      continue
51                  elif caracter == ' ':
52                      self.columna += 1
53                      texto = texto[1:]
54                      self.estadoAnterior = 'A'
55                      self.estadoActual = 'A'
56                      continue
57                  elif caracter == '\t':
58                      self.columna += 4
```

```
58                  self.columna += 4
59                  texto = texto[1:]
60                  self.estadoAnterior = 'A'
61                  self.estadoActual = 'A'
62                  continue
63              else:
64                  validandoError = True
65                  self.almacenarError(caracter,'invalido')
66          # valida cuando hay comentarios de una Linea
67          elif self.estadoActual == 'D':
68              if caracter == '-':
69                  self.almacenarToken(caracter,'guion')
70                  self.estadoAnterior = 'D'
71                  self.estadoActual = 'G'
72              else:
73                  validandoError = True
74                  self.almacenarError(caracter,'invalido falta -')
75          elif self.estadoActual == 'G':
76              if caracter == '-':
77                  self.almacenarToken(caracter,'guion')
78                  self.estadoAnterior = 'G'
79                  self.estadoActual = 'J'
80              else:
81                  validandoError = True
82                  self.almacenarError(caracter,'invalido falta -')
83          elif self.estadoActual == 'J':
84              if caracter.lower() in self.letrasComentarios:
85                  tok += caracter
86                  self.estadoAnterior = 'J'
87                  self.estadoActual = 'J'
88              elif caracter == ' ':
89                  if tok != '':
90                      self.almacenarToken(tok,'comentario')
91                  tok = ''
92                  self.estadoAnterior = 'J'
93                  self.estadoActual = 'J'
94                  self.columna += 1
```

El código continúa hasta aumentar la columna, eliminar el carácter analizado de la cadena de texto y retornar el estado actual si corresponde al estado de aceptación.

```
587             self.columna += 1
588             texto = texto[1:]
589         return self.estadoActual in self.estadoFinal
```

Método almacenarToken

Recibe como parámetro el lexema y la descripción del token y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en el automata, luego este se almacena en la tabla de tokens.

```
904     #Metodos para almacenar token lexicos
905     def almacenarToken(self, lexema,t):
906         newToken = Token(self.fila, self.columna, lexema,t)
907         self.tabla.append(newToken)
```

Método almacenarError

Recibe como parámetro el lexema y la descripción del token y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en el autómata, luego este se almacena en la tabla de errores.

```
909     def almacenarError(self, lexemaError,t):
910         newToken1 = Token(self.fila, self.columna, lexemaError,t)
911         self.tablaErrores.append(newToken1)
```

Método analizadorSintactico

Este método sirve para obtener los datos de la tabla de tokens y validar que se respete la estructura de las funciones y de los comentarios, por lo que se utiliza un bucle while para recorrer toda la tabla. En las validaciones if podrá observar que se va validando las opciones para las asignaciones de los valores que corresponden a sus determinados campos. Esto es algo que se repite en todo el método, por lo cual solo se agregan las otras imágenes.

```
591     def analizadorSintactico(self):
592         estadoAct = 'S'
593         estadoAnt = ''
594         it = 0
595         while it < len(self.tabla):
596             if estadoAct == 'S':
597                 if self.tabla[it].lexema in self.reservadasFunciones:
598                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'funcion')
599                     estadoAnt = 'S'
600                     estadoAct = 'A'
601                 elif self.tabla[it].lexema == '/':
602                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'comentario')
603                     estadoAnt = 'A'
604                     estadoAct = 'M'
605                 elif self.tabla[it].lexema == '-':
606                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'comentario')
607                     estadoAnt = 'A'
608                     estadoAct = 'P'
609                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ' or self.tabla[it].lexema == '\t':
610                     estadoAnt = 'S'
611                     estadoAct = 'S'
612                 else:
613                     self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
614             elif estadoAct == 'A':
615                 if self.tabla[it].lexema != '':
616                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'variable')
617                     estadoAnt = 'A'
618                     estadoAct = 'B'
619                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
620                     estadoAnt = 'A'
621                     estadoAct = 'A'
622                 else:
623                     self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
624             elif estadoAct == 'B':
625                 if self.tabla[it].lexema == '=':
626                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'igual')
627                     estadoAnt = 'B'
```

```
624             elif estadoAct == 'B':
625                 if self.tabla[it].lexema == '=':
626                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'igual')
627                     estadoAnt = 'B'
628                     estadoAct = 'C'
629                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
630                     estadoAnt = 'B'
631                     estadoAct = 'B'
632                 else:
633                     self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
634             elif estadoAct == 'C':
635                 if self.tabla[it].lexema.lower() == 'nueva':
636                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'reservada')
637                     estadoAnt = 'C'
638                     estadoAct = 'D'
639                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
640                     estadoAnt = 'C'
641                     estadoAct = 'C'
642                 else:
643                     self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
644             elif estadoAct == 'D':
645                 if self.tabla[it].lexema in self.reservadasFunciones:
646                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'funcion')
647                     estadoAnt = 'D'
648                     estadoAct = 'E'
649                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
650                     estadoAnt = 'D'
651                     estadoAct = 'D'
652                 else:
653                     self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
654             elif estadoAct == 'E':
655                 if self.tabla[it].lexema == '(':
656                     self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'parentesis')
657                     estadoAnt = 'E'
658                     estadoAct = 'F'
659                 elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
660                     estadoAnt = 'E'
```

```
624 >       elif estadoAct == 'B': ···
634 >       elif estadoAct == 'C': ···
644 >       elif estadoAct == 'D': ···
654 v       elif estadoAct == 'E':
655 v           if self.tabla[it].lexema == '(':
656                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'parentesis')
657                 estadoAnt = 'E'
658                 estadoAct = 'F'
659 v           elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
660                 estadoAnt = 'E'
661                 estadoAct = 'E'
662 v           else:
663                 self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
664 v       elif estadoAct == 'F':
665 v           if self.tabla[it].lexema == ')':
666                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'parentesis')
667                 estadoAnt = 'F'
668                 estadoAct = 'W'
669 v           elif self.tabla[it].lexema == '“':
670                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'comilla')
671                 estadoAnt = 'F'
672                 estadoAct = 'H'
673 v           elif self.tabla[it].lexema == '”':
674                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'comilla')
675                 estadoAnt = 'F'
676                 estadoAct = 'F'
677 v           elif self.tabla[it].lexema == ',':
678                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'coma')
679                 estadoAnt = 'F'
680                 estadoAct = 'F'
681 v           elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
682                 estadoAnt = 'F'
683                 estadoAct = 'F'
684 v           else:
685                 self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'invalido')
686 >       elif estadoAct == 'H': ···
700 >       elif estadoAct == 'W': ···
```

```
744 v       elif estadoAct == 'P':
745 v           if self.tabla[it].lexema == '-':
746                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'guion -')
747                 estadoAnt = 'P'
748                 estadoAct = 'P'
749 v           elif self.tabla[it].lexema != '':
750                 self.almacenarSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'caracter')
751                 estadoAnt = 'P'
752                 estadoAct = 'A'
753 v           elif self.tabla[it].lexema == '\n' or self.tabla[it].lexema == ' ':
754                 estadoAnt = 'P'
755                 estadoAct = 'P'
756 v           else:
757                 self.almacenarErrorSintactico(self.tabla[it].fila, self.tabla[it].columna, self.tabla[it].lexema,'- ó caracter invalido')
758       it +=1
```

Método escribiendoArchivo

Este método básicamente realiza lo mismo que el método anterior, solamente que las validaciones no almacenan nada ya que solamente se requiere que se reconozca los datos para poder convertirlos a sentencias mongoDB, al final retorna la cadena donde se almacenaron las sentencias.

```
760     def escribiendoArchivo(self):
761         estadoAct = 'S'
762         estadoAnt = ''
763         self.sentencias = ''
764         compararFuncion = ''
765         nameColeccion = ''
766         archivoJson = ''
767         it = 0
768         while it < len(self.tablaSintactico):
769             if estadoAct == 'S':
770                 if self.tablaSintactico[it].lexema in self.reservadasFunciones:
771                     compararFuncion = self.tablaSintactico[it].lexema
772                     estadoAnt = 'S'
773                     estadoAct = 'A'
774             elif estadoAct == 'A':
775                 if self.tablaSintactico[it].lexema != '':
776                     estadoAnt = 'A'
777                     estadoAct = 'B'
778                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
779                     estadoAnt = 'A'
780                     estadoAct = 'A'
781             elif estadoAct == 'B':
782                 if self.tablaSintactico[it].lexema == '=':
783                     estadoAnt = 'B'
784                     estadoAct = 'C'
785                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
786                     estadoAnt = 'B'
787                     estadoAct = 'B'
788             elif estadoAct == 'C':
789                 if self.tablaSintactico[it].lexema.lower() == 'nueva':
790                     estadoAnt = 'C'
791                     estadoAct = 'D'
792                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
793                     estadoAnt = 'C'
794                     estadoAct = 'C'
```

```
795             elif estadoAct == 'D':
796                 if self.tablaSintactico[it].lexema in self.reservadasFunciones:
797                     compararFuncion = self.tablaSintactico[it].lexema
798                     estadoAnt = 'D'
799                     estadoAct = 'E'
800                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
801                     estadoAnt = 'D'
802                     estadoAct = 'D'
803             elif estadoAct == 'E':
804                 if self.tablaSintactico[it].lexema == '(':
805                     estadoAnt = 'E'
806                     estadoAct = 'F'
807                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
808                     estadoAnt = 'E'
809                     estadoAct = 'E'
810             elif estadoAct == 'F':
811                 if self.tablaSintactico[it].lexema == ')':
812                     estadoAnt = 'F'
813                     estadoAct = 'W'
814                 elif self.tablaSintactico[it].lexema == '“':
815                     estadoAnt = 'F'
816                     estadoAct = 'H'
817                 elif self.tablaSintactico[it].lexema == '”':
818                     #self.sentencias += '”'
819                     estadoAnt = 'F'
820                     estadoAct = 'F'
821                 elif self.tablaSintactico[it].lexema == ',':
822                     estadoAnt = 'F'
823                     estadoAct = 'I'
824                 elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
825                     estadoAnt = 'F'
826                     estadoAct = 'F'
```

```
827                 elif estadoAct == 'H':
828                     if self.tablaSintactico[it].lexema == '”':
829                         #self.sentencias += ''
830                         estadoAnt = 'H'
831                         estadoAct = 'F'
832                     elif self.tablaSintactico[it].lexema != '”':
833                         nameColeccion = str(self.tablaSintactico[it].lexema)
834                         estadoAnt = 'H'
835                         estadoAct = 'H'
836                     elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
837                         estadoAnt = 'H'
838                         estadoAct = 'H'
839                 elif estadoAct == 'I':
840                     if self.tablaSintactico[it].lexema == ')':
841                         estadoAnt = 'I'
842                         estadoAct = 'W'
843                     elif self.tablaSintactico[it].lexema == '“':
844                         estadoAnt = 'I'
845                         estadoAct = 'J'
846                     elif self.tablaSintactico[it].lexema == '”':
847                         #self.sentencias += ''
848                         estadoAnt = 'I'
849                         estadoAct = 'I'
850                     elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
851                         estadoAnt = 'I'
852                         estadoAct = 'I'
853                 elif estadoAct == 'J':
854                     if self.tablaSintactico[it].lexema == '”':
855                         #self.sentencias += ''
856                         estadoAnt = 'J'
857                         estadoAct = 'I'
858                     elif self.tablaSintactico[it].lexema != '”':
859                         archivoJson += str(self.tablaSintactico[it].lexema)
860                         estadoAnt = 'J'
861                         estadoAct = 'J'
862                     elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
863                         estadoAnt = 'J'
```

```
862             elif self.tablaSintactico[it].lexema == '\n' or self.tablaSintactico[it].lexema == ' ':
863                 estadoAnt = 'J'
864                 estadoAct = 'J'
865         elif estadoAct == 'W':
866             if self.tablaSintactico[it].lexema == ';':
867                 estadoAnt = 'W'
868                 estadoAct = 'S'
869                 if compararFuncion.strip() == 'CrearBD':
870                     self.sentencias += 'use(nombreBaseDatos);'
871                 elif compararFuncion.strip() == 'EliminarBD':
872                     self.sentencias += 'db.dropDatabase();'
873                 elif compararFuncion.strip() == 'CrearColeccion':
874                     self.sentencias += 'db.createCollection('+nameColeccion+');'
875                 elif compararFuncion.strip() == 'EliminarColeccion':
876                     self.sentencias += 'db.'+ nameColeccion+ '.drop();'
877                 elif compararFuncion.strip() == 'InsertarUnico':
878                     self.sentencias += 'db.'+ nameColeccion+ '.insertOne(“'+archivoJson+'”);'
879                 elif compararFuncion.strip() == 'ActualizarUnico':
880                     self.sentencias += 'db.'+ nameColeccion+ '.updateOne(“'+archivoJson+'”);'
881                 elif compararFuncion.strip() == 'EliminarUnico':
882                     self.sentencias += 'db.'+ nameColeccion+ '.deleteOne(“'+archivoJson+'”);'
883                 elif compararFuncion.strip() == 'BuscarTodo':
884                     self.sentencias += 'db.'+ nameColeccion+ '.find();'
885                 elif compararFuncion.strip() == 'BuscarUnico':
886                     self.sentencias += 'db.'+ nameColeccion+ '.findOne();'
887                 self.sentencias += '\n\n'
888                 archivoJson = ''
889                 nameColeccion = ''
890 
891             it +=1
892         archivoMongoDB = str(self.sentencias)
893         return archivoMongoDB
```

Método almacenarSintactico

Recibe como parámetro el dato y la descripción del token y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en la GLC, luego este se almacena en la tabla de tokens.

```
895     #Metodos para almacenar Sintacticos
896     def almacenarSintactico(self,fila, columna, dato, t):
897         newSin = Token(fila, columna, dato, t)
898         self.tablaSintactico.append(newSin)
```

Método almacenarErrorSintactico

Recibe como parámetro el dato y la descripción del token y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en la GLC, luego este se almacena en la tabla de errores.

```
900     def almacenarErrorSintactico(self,fila, columna, dato, t):
901         newESin = Token(fila, columna, dato, t)
902         self.tablaErroresSintacticos.append(newESin)
```

Métodos para obtener las listas

Estos métodos sirven para retornar cada una de las tablas y así poder obtenerlas en las ventanas para recorrerlas y mostrarlas en las tablas.

```
913     #Metodos para obtener las tablas
914     def obtenerTablaTokens(self):
915         return self.tabla
916 
917     def obtenerTablaErrores(self):
918         return self.tablaErrores
919 
920     def obtenerTablaSintactico(self):
921         return self.tablaSintactico
922 
923     def obtenerTablaSintacticoErrores(self):
924         return self.tablaErroresSintacticos
```

Métodos para validar listas vacías

Estos métodos sirven para retornar True o False de acuerdo a si la lista se encuentra vacía o no, esto servirá para validar si es posible escribir el archivo con las sentencias en mongoDB, ya que si existen datos en las listas de errores, entonces no se ejecutará el método de escribirArchivo.

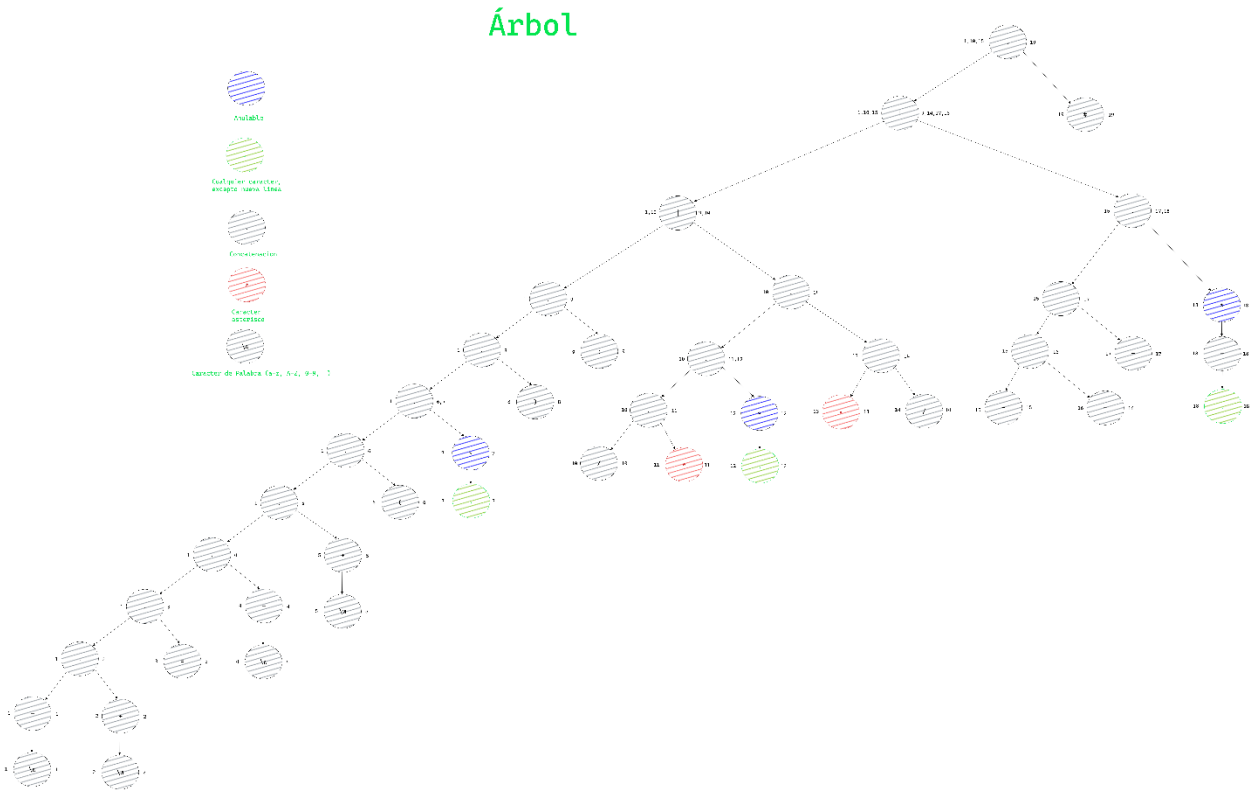
```
946     def vacioTablaErrorLexico(self):
947         if len(self.tablaErrores) == 0:
948             #Si está vacio retorna True
949             return True
950         else:
951             return False
952 
953     def vacioTablaErrorSintactico(self):
954         if len(self.tablaErroresSintacticos) == 0:
955             #Si está vacio retorna True
956             return True
957         else:
958             return False
```

Autómata Finito Determinista

En la imagen siguiente se ilustra el autómata finito determinista que fue diseñado para darle solución a la lectura correcta del archivo a analizar en el software.

El AFD fue construido a partir del método del árbol, tomando em cuenta la expresión regular.

```
(\w+\s+\w+\s+=\s+nueva\s+\w+\(.*\);)|(\/\*(.)*\*\/)|((---)(.+)*)
```



Para concatenacion

Elemento	Follow
1- \w	2
2- \w	3
3- =	4
4- \w	5
5- \w	6
6- (	7,8
7- .	8
8-)	9
9- ;	19
10- /	11
11- *	12,13
12- .	13
13- *	14
14- /	19
15- -	16
16- -	17
17- -	18,19
18- .	19
19- #	---------

Para * +

Elemento	Follow
1- \w	1,2
2- \w	2,3
3- =	4
4- \w	4,5
5- \w	5,6
6- (	7,8
7- .	7,8
8-)	9
9- ;	19
10- /	11
11- *	12,13
12- .	12,13
13- *	14
14- /	19
15- -	16
16- -	17
17- -	18,19
18- .	18,19
19- #	---------

Tabla de follow

Elemento	Follow
1- \w	1,2
2- \w	2,3
3- =	4
4- \w	4,5
5- \w	5,6
6- (	7,8
7- .	7,8
8-)	9
9- ;	19
10- /	11
11- *	12,13
12- .	12,13
13- *	14
14- /	19
15- -	16
16- -	17
17- -	18,19
18- .	18,19
19- #	----------

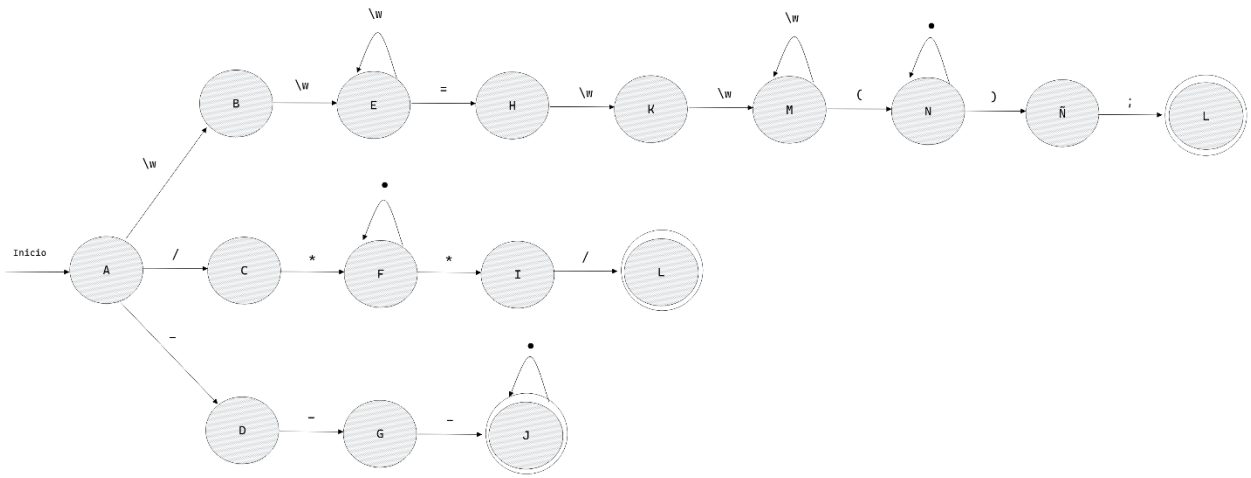
Tabla de Transición

	\w	=	(	)	.	;	*	/	-
A= 1,10,15	B = 1,2	---------	---------	---------	---------	---------	---------	C = 11	D = 16
B = 1,2	E = 1,2,3	---------	---------	---------	---------	---------	---------	---------	---------
C = 11	---------	---------	---------	---------	---------	---------	F = 12,13	---------	---------
D = 16	---------	---------	---------	---------	---------	---------	---------	---------	G = 17
E = 1,2,3	E = 1,2,3	H = 4	---------	---------	---------	---------	---------	---------	---------
F = 12,13	---------	---------	---------	---------	F = 12,13	---------	I = 14	---------	---------
G = 17	---------	---------	---------	---------	---------	---------	---------	---------	J = 18,19
H = 4	K = 4,5	---------	---------	---------	---------	---------	---------	---------	---------
I = 14	---------	---------	---------	---------	---------	---------	---------	L = 19	---------
J = 18,19	---------	---------	---------	---------	J = 18,19	---------	---------	---------	---------
K = 4,5	M = 4,5,6	---------	---------	---------	---------	---------	---------	---------	---------
M = 4,5,6	M = 4,5,6	---------	N = 7,8	---------	---------	---------	---------	---------	---------
N = 7,8	---------	---------	---------	Ñ = 9	N = 7,8	---------	---------	---------	---------
Ñ = 9	---------	---------	---------	---------	---------	L = 19	---------	---------	---------

Tabla de Transición

	\w	=	(	)	.	;	*	/	-
A= 1,10,15	B	----	----	----	----	----	----	C	D
B = 1,2	E	----	----	----	----	----	----	----	----
C = 11	----	----	----	----	----	----	F	----	----
D = 16	----	----	----	----	----	----	----	----	G
E = 1,2,3	E	H	----	----	----	----	----	----	----
F = 12,13	----	----	----	----	F	----	I	----	----
G = 17	----	----	----	----	----	----	----	----	J
H = 4	K	----	----	----	----	----	----	----	----
I = 14	----	----	----	----	----	----	----	L	----
J = 18,19	----	----	----	----	J	----	----	----	----
K = 4,5	M	----	----	----	----	----	----	----	----
M = 4,5,6	M	----	N	----	----	----	----	----	----
N = 7,8	----	----	----	Ñ	N	----	----	----	----
Ñ = 9	----	----	----	----	----	L	----	----	----

AUTOMATA FINITO DETERMINISTA



Gramática Libre del Contexto (GLC)

La GLC fue construida para verificar el orden de las funciones y los comentarios que se obtienen y que se encuentran en una tabla de tokens, la cual fue analizada por el analizador léxico.

```
Funcion Nombre = nueva Funcion();
Funcion Nombre = nueva Funcion(“identificador”);
Funcion Nombre = nueva Funcion(””, “JSON”);
/*
Comentario de
varias líneas
*/
--- comentario
```

```
Simbolo de Inicio
     {S}

Terminales
funcion :                f
nombre de variable:      v
palabra reservada nueva: n
letras:                  l (para identificador y json)
caracteres:              c (cualquier caracter para comentarios)
{f,v,/,-,=,n,(,),l,,,;,*,c,“,”}

No terminales
{S,A,B,C,D,E,F,H,W,M,N,U,P,Z}
```

Producciones

$$
\begin{aligned}
S &\longrightarrow fA \\
A &\longrightarrow vB \\
S &\longrightarrow /M \\
S &\longrightarrow -P \\
B &\longrightarrow =C \\
C &\longrightarrow nD \\
D &\longrightarrow fE \\
E &\longrightarrow (F \\
F &\longrightarrow)W \\
F &\longrightarrow “H”F \\
H &\longrightarrow l \\
H &\longrightarrow ”F \\
F &\longrightarrow ,F \\
W &\longrightarrow ;Z \\
M &\longrightarrow *N \\
N &\longrightarrow cN \\
N &\longrightarrow *U \\
U &\longrightarrow /Z \\
P &\longrightarrow -PcZ \\
Z &\longrightarrow \varepsilon
\end{aligned}
$$