UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTADA DE INGENIERIA

ESCUELA DE CIENCIAS Y SISTEMAS

LENGUAJES FORMALES Y DE PROGRAMACION

PRIMER SEMESTRE 2,023

Manual Técnico

Autor: Brayan Estiben Micá Pérez

201907343

**Introducción**

El presente manual se presentan las especificaciones técnicas del programa de carga de archivos con extensión. .txt, se muestra cada función por separado para poder hacer modificaciones al código posteriormente.

**Objetivo**

* Otorgar soporte técnico a los desarrolladores para poder modificar la aplicación.

**Requerimientos**

* Equipo: Dual Core o superior
* Sistema Operativo: Windows 7 o superior
* Memoria RAM: 1 Gb mínimo o superior
* Resolución gráfica: mínimo 800\*600 o superior
* Tener instalado Python 3 con todas sus librerías
* Tener instalado el gestor de paquetes pip para descargar extensiones

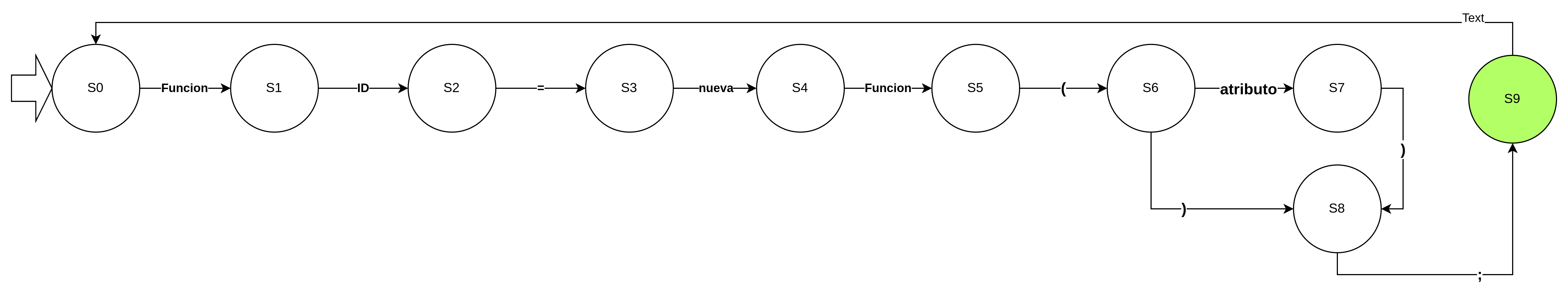
**INDICE DEL MANUAL**

1. Gramáticas
2. Opciones del menú archivo
3. Mostrar Posición x,y del apuntador
4. Opción Analizar
5. Opción Tokens
6. Opción errores
7. **Gramáticas:**

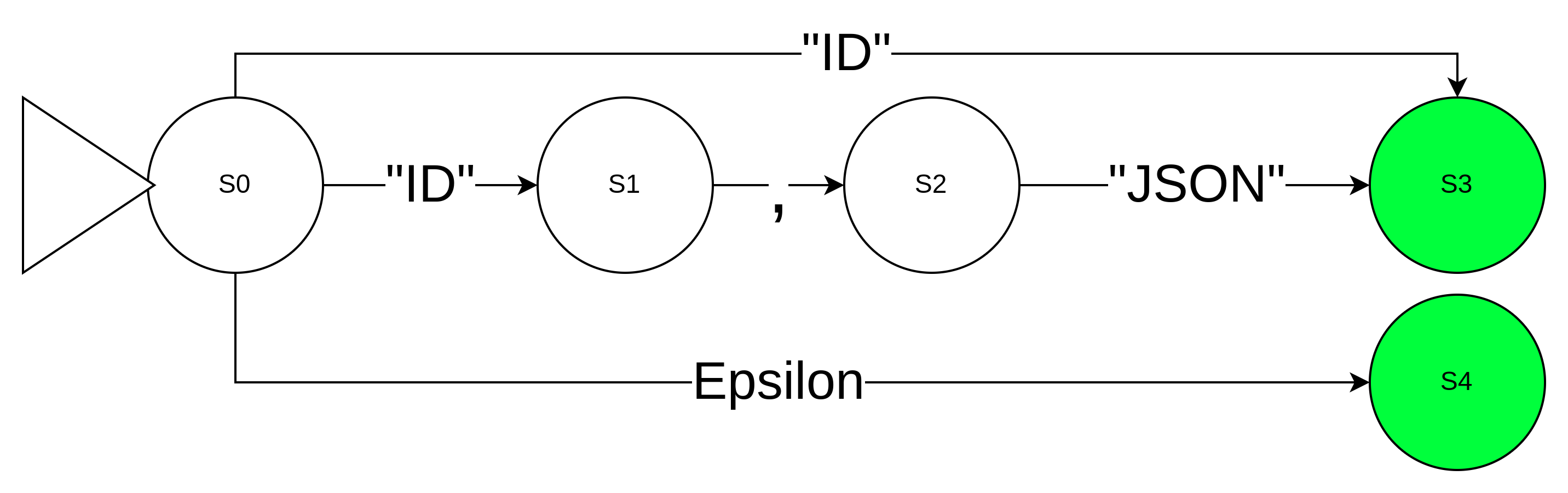
|  |  |
| --- | --- |
| Análisis de una función  Funcion . ID . = . nueva . Funcion . ( atributo?) . ; | S0 -> Funcion S1  S1 -> ID S2  S2 -> = S3  S3 -> nueva S4  S4 -> Funcion S5  S5 -> ( S6  S6 -> atributo S7  | ) S8  S7 -> ) S8  S8 -> ; S0  | ; |
| Analisis de Atributo  Atributo : ("ID" | "ID".,."Json")? | SO-> "ID" S1  | “ID” S3  | Epsilon S4  S1 -> , S2  S2 -> “JSON” S3 |
| Analisis de JSON  {."ID".:."ID".(,."ID".:."ID")\*}.(,.{.$set.:.{."ID".:."ID".}.})? | S0 -> { S1  S1 -> “ID” S2  S2 -> : S3  S3 -> “ID” S4  S4 -> , S5  | } S9  S5 -> “ID” S6  S6 -> : S7  S7 -> “ID” S8  S8 -> } S9  | , S5  S9 - > , S10  S10 -> { S11  S11 -> $set S12  S12 -> : S13  S13 -> { S14  S14 -> “ID”  S15 -> : S16  S16 -> “ID” S17  S17 -> } S18  S18 -> } S19 |

**Autómata:**

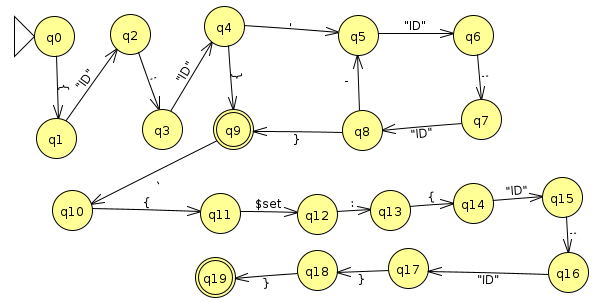
Función:



Atributo:

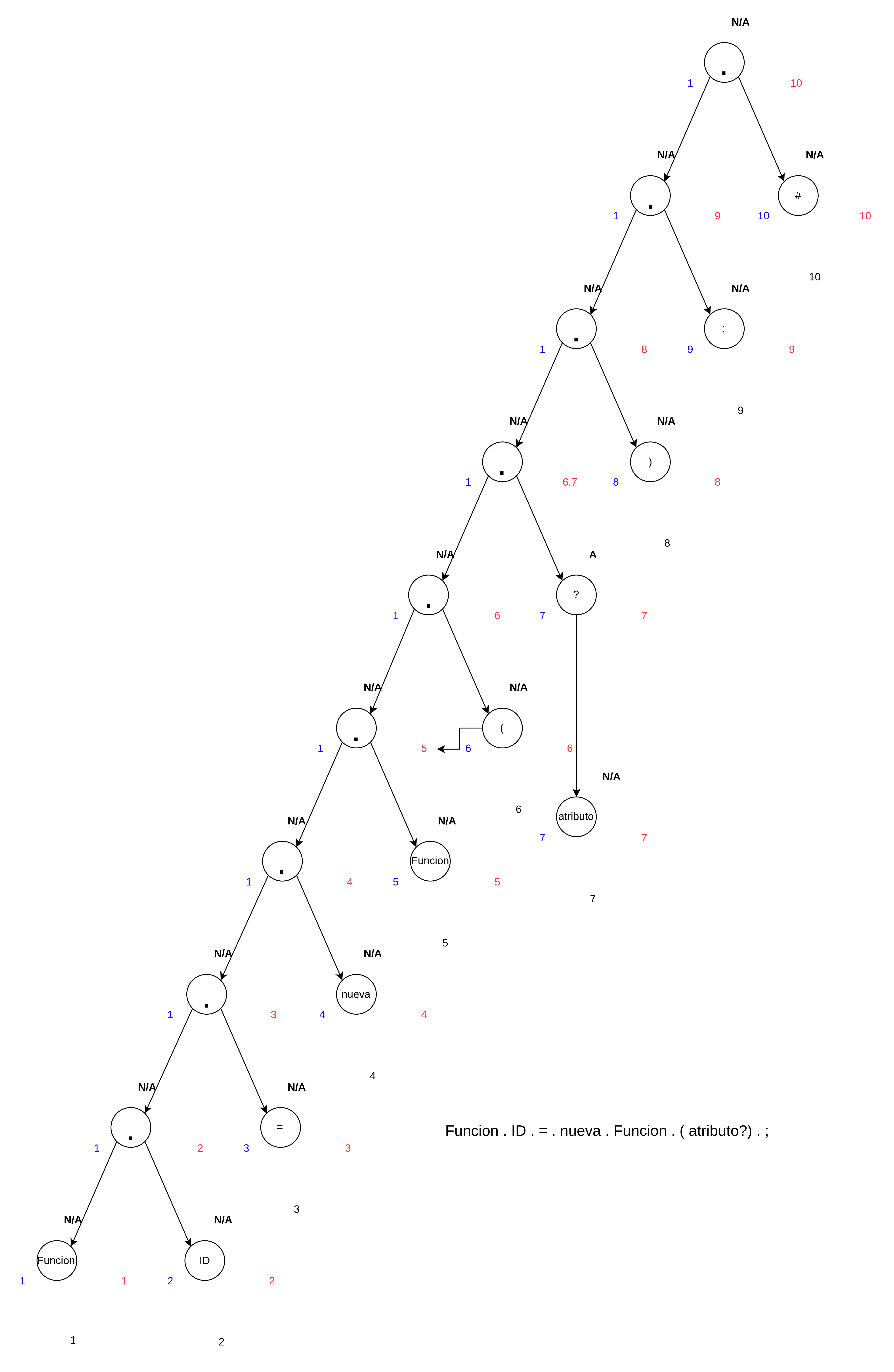


JSON:

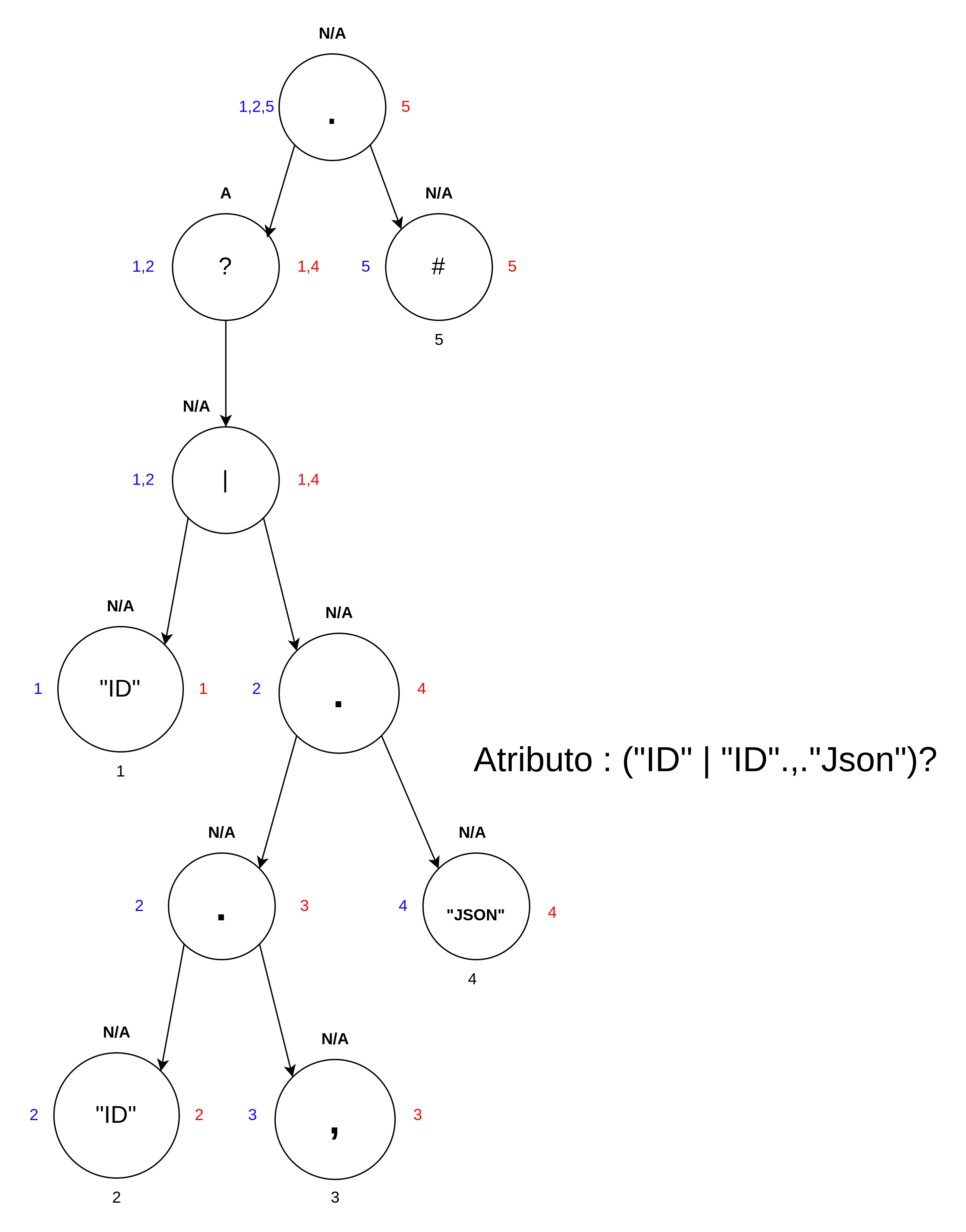


**Diagrama del Árbol:**

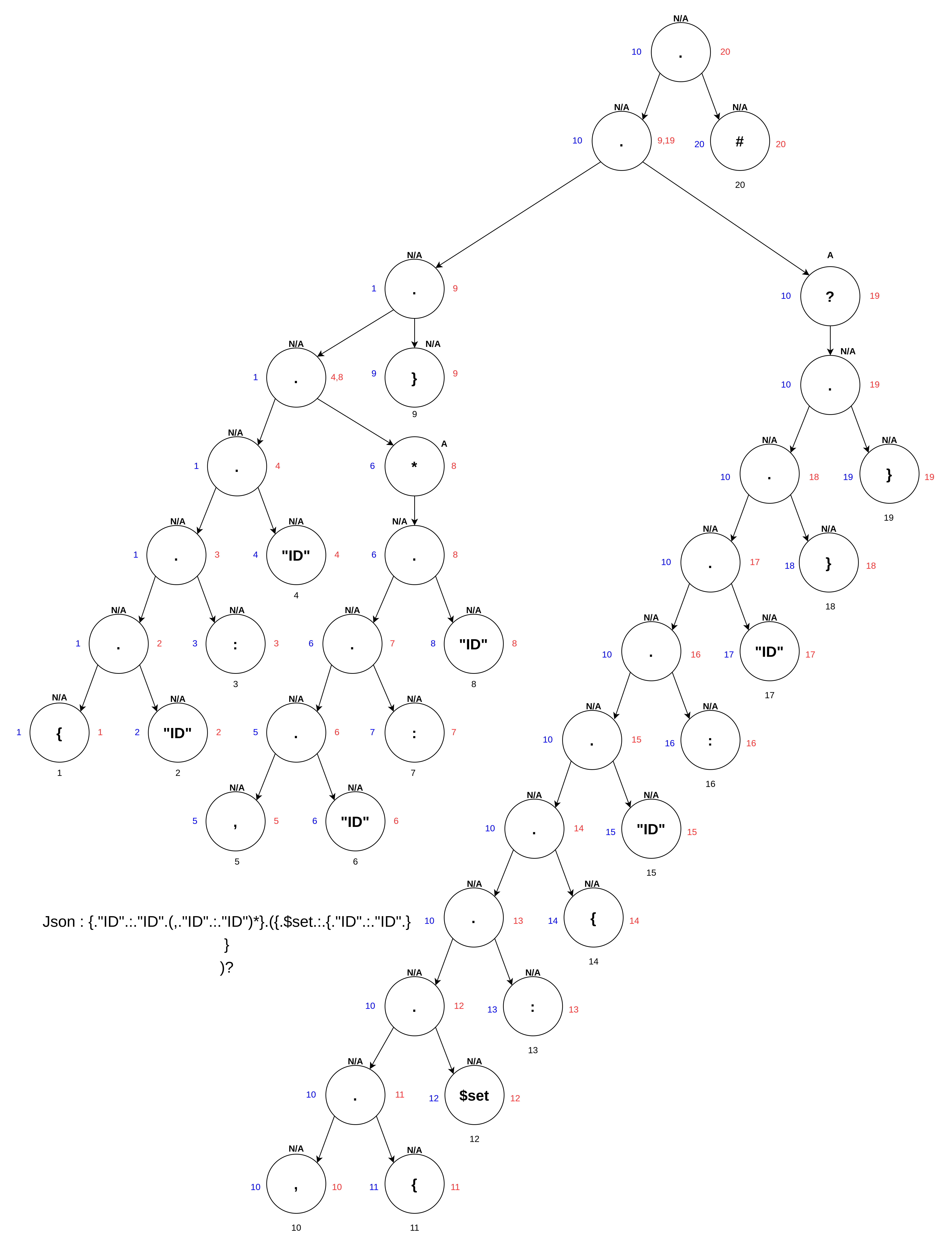
Funcion



Atributo.



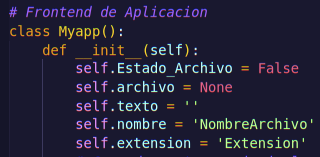
JSON



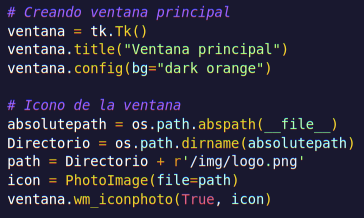
1. **Opciones del menú archivo**

La función Myapp es el constructor de la clase que se encarga de crear el frontend de la aplicación.

Primero, se definen algunas variables de instancia para controlar el estado del archivo, la ruta del archivo, el texto cargado en memoria, el nombre y la extensión del archivo.

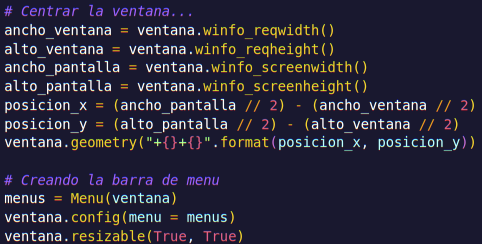


Luego, se crea una ventana principal utilizando el método Tk() del módulo tkinter. Se establece el título de la ventana con el método title, se configura el color de fondo con el método config, y se define el icono de la ventana utilizando el método wm\_iconphoto.

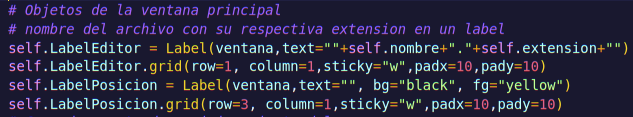


A continuación, se centra la ventana utilizando información obtenida de los métodos winfo\_reqwidth, winfo\_reqheight, winfo\_screenwidth, y winfo\_screenheight.

Se crea una barra de menú utilizando el objeto Menu y se asocia a la ventana principal con el método config(menu = menus). También se habilita la capacidad de redimensionar la ventana utilizando el método resizable.

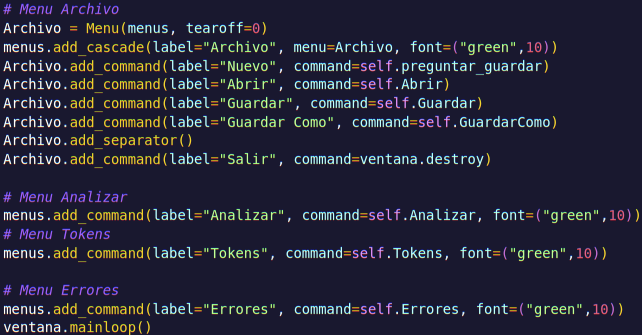


Se crean varios objetos de la ventana principal, como un label para mostrar el nombre del archivo actual, un label para mostrar la posición actual del cursor, y un cuadro de edición para permitir al usuario escribir o abrir archivos.



Por último, se crean tres menús (Archivo, Analizar y Errores) utilizando el objeto Menu y se agregan a la barra de menú con el método add\_cascade para el menú Archivo y add\_command para los menús Analizar y Errores. Cada uno de estos menús está asociado a una función específica que se encargará de realizar la tarea correspondiente.

Finalmente, se ejecuta el loop principal de la aplicación con el método mainloop.



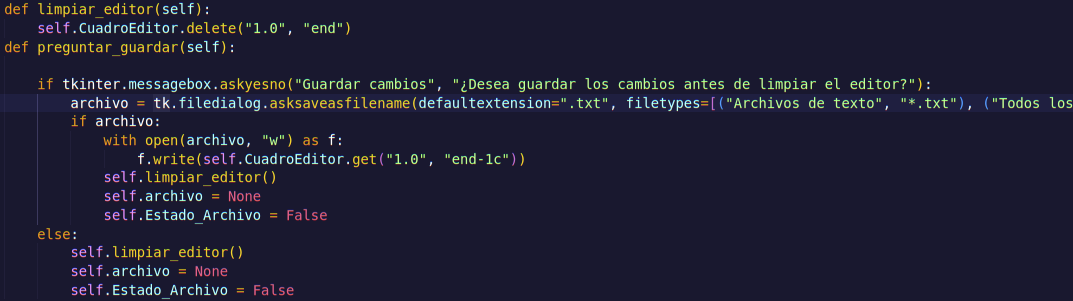
La opción de menu -> archivo-> abrir, verifica la extensión del archivo asi como la selección del mismo además se tiene una función que recupera dicha respuesta si el archivo es seleccionado o no.



El método preguntar\_guardar muestra un mensaje en una ventana de diálogo preguntándole al usuario si desea guardar los cambios antes de limpiar el editor.

Si el usuario responde sí, entonces el código muestra otro cuadro de diálogo que le permite al usuario seleccionar un archivo para guardar los cambios. Si el usuario selecciona un archivo, el contenido del cuadro de texto es guardado en dicho archivo utilizando la función open de Python con el modo "w", lo que significa que se abrirá para escribir en él. Luego, el método limpiar\_editor se llama para borrar el contenido del cuadro de texto, y las variables archivo, Estado\_Archivo y self.archivo se resetean.

Si el usuario responde no a la ventana de diálogo, entonces simplemente se llama al método limpiar\_editor, y las variables archivo, Estado\_Archivo y self.archivo también se resetean.



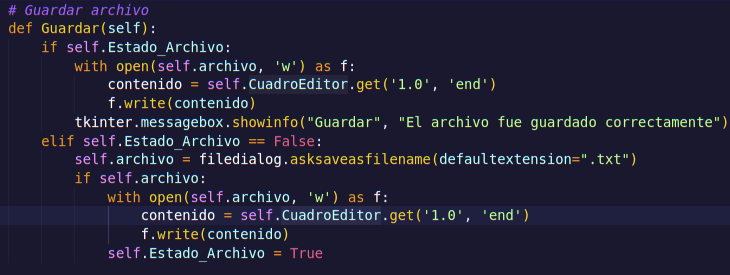
La función Guardar se encarga de guardar el contenido del cuadro de texto en un archivo.

Primero, comprueba si la variable Estado\_Archivo es verdadera o falsa para determinar si el archivo ya está guardado o no. Si ya se guardó anteriormente, entonces el código abre el archivo utilizando la instrucción with open(self.archivo, 'w') as f, donde self.archivo es un string que contiene la ruta y nombre del archivo guardado anteriormente, y el modo 'w' indica que el archivo se abrirá en modo escritura.

A continuación, se extrae el contenido del cuadro de texto utilizando el método get del objeto CuadroEditor. Este método toma dos argumentos, "1.0" y "end", que indican que se obtendrá todo el texto desde la primera fila hasta la última. Luego, el contenido se escribe en el archivo utilizando el método write del objeto f.

Si el archivo no se ha guardado previamente (Estado\_Archivo == False), entonces se muestra una ventana para que el usuario seleccione la ubicación y el nombre del archivo en el que desea guardar el contenido del cuadro de texto. Si el usuario selecciona un archivo, el código realiza las mismas operaciones que si el archivo ya hubiera sido guardado para guardar el contenido en el archivo.

Finalmente, si el contenido se guarda correctamente, se muestra un mensaje en una ventana emergente diciendo que el archivo se guardó correctamente, y la variable Estado\_Archivo se actualiza a verdadero para indicar que el archivo ahora se ha guardado.



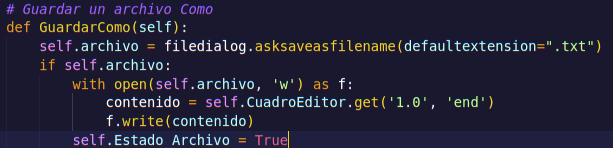
La función GuardarComo se encarga de guardar el contenido del cuadro de texto en un nuevo archivo cuya ruta y nombre serán seleccionados por el usuario.

La función comienza abriendo una ventana emergente utilizando el método asksaveasfilename del módulo filedialog. Este método muestra una ventana para que el usuario seleccione la ubicación y el nombre del archivo en el que desea guardar el contenido. La opción defaultextension=".txt" indica que, si el usuario no especifica una extensión para el archivo, se agregará automáticamente la extensión .txt al final del nombre del archivo.

Si el usuario efectivamente selecciona un archivo, el código continúa ejecutando el resto de la función. Primero, el código abre el archivo en modo de escritura utilizando la instrucción with open(self.archivo, 'w') as f, donde self.archivo contiene la ruta y nombre del archivo seleccionado por el usuario.

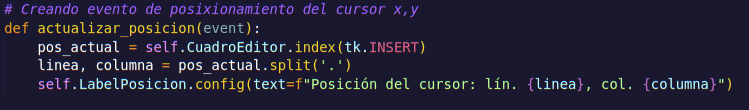
A continuación, se extrae el contenido del cuadro de texto de la misma forma que en la función Guardar anterior, utilizando el método get del objeto CuadroEditor. El contenido se escribe en el archivo utilizando el método write del objeto f.

Finalmente, se actualiza la variable Estado\_Archivo a verdadero para indicar que el archivo ha sido guardado correctamente.



1. **Mostrar Posición x,y del apuntador**

Se define una función llamada actualizar\_posicion que se encarga de actualizar el label de la posición del cursor cada vez que el usuario presiona una tecla en el cuadro de edición. Esta función se vincula al evento <KeyRelease> del cuadro de edición utilizando el método bind.



1. **Opción Analizar**
2. **Opción Tokens**
3. **Opción de Errores.**