

T3-MUSIK

MANUAL TECNICO

LENGUAJES FORMALES Y DE
PROGRAMACION



SECCIÓN: N

PROYECTO 1

RODRIGO ALEJANDRO HERNÁNDEZ
DE LEÓN

CONTENIDO

I. Introducción	2
II. Objetivos	2
III. Dirigido.....	2
IV. Especificación técnica	3
1. Requisitos de Hardware.....	3
2. Requisitos de Software.....	3
V. Lógica del programa	4
1. Descripción del lenguaje .mt3.....	4
2. Autómata grafico	5
3. Expresiones Regulares de los tokens	5
4. Código fuente:.....	6
Método Analizar	6
Función aslistacan:.....	6
Clase Token.....	7
Clase Error	7
Clase Canción:.....	8
VI. Créditos	9

I. Introducción

El Programa de T3-Musik es un reproductor de música realizado en el lenguaje de programación Python donde en este manual se explicara como se realizó el programa junto con las funciones utilizadas para que diera como resultado este reproductor por medio de un análisis léxico realizado con un autómata finito determinista.

II. Objetivos

El objetivo primordial de este manual es ayudar a los distintos programadores y auxiliares de cátedra de los cursos de ciencias de la computación a informarse y guiarse de la comprensión del desarrollo de la aplicación hecha por el creador de este reproductor de música

El objetivo de este sistema es poder comprender la utilidad del uso de autómatas finitos para la lectura de un archivo donde pueden abstraerse sus atributos y darle la funcionalidad que requiere como en este caso de reproductor de música para entender la funcionalidad de este analizador léxico.

III. Dirigido

Este manual esta orientado a todos los distintos programadores interesados en el desarrollo de esta aplicación y al auxiliar de cátedra del curso de lenguajes formales quienes tienen el conocimiento acerca de estos temas.

IV. Especificación técnica

1.Requisitos de Hardware

- Computadora de escritorio o portátil.
- Mínimo 4GB de Memoria RAM.
- 250GB de Disco Duro o Superior.
- Procesador Inter Core i3 o superior.
- Resolución gráfica mínimo 1024 x 768 pixeles.

2.Requisitos de Software

- Sistema Operativo Windows 7 o superior
- Python en la versión 3.9.2
- Visual Studio Code u otro editor de texto.
- Librería externa PILLOW
- Navegador web para la lectura de archivos HTML

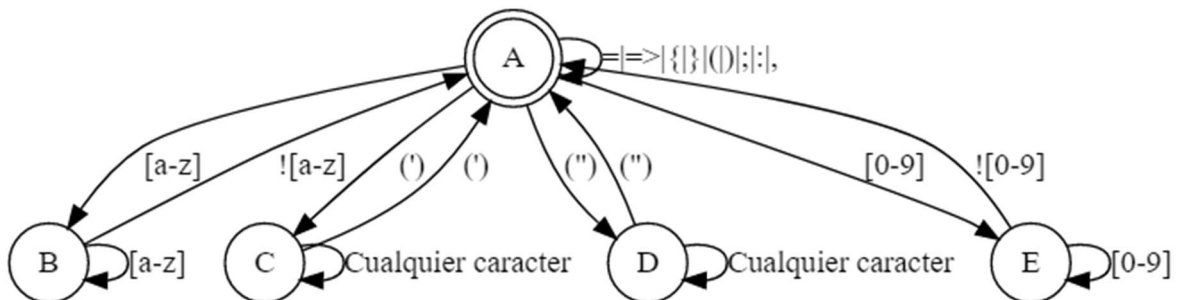
V. Lógica del programa

1. Descripción del lenguaje .mt3

Token	Descripción	Lexema
replist	Palabra reservada	replist
igual	Carácter igual	=
flecha	Secuencia de un carácter igual seguido de un carácter mayor que	=>
parentesis	Carácter paréntesis que abre	(
parentesis	Carácter paréntesis que cierra)
dos puntos	Carácter dos puntos	:
punto y coma	Carácter punto y coma	;
coma	Carácter coma	,
llave	Carácter llave que abre	{
llave	Carácter llave que cierra	}
cadena	Cadena de caracteres que empieza y termina con el carácter comilla doble o comilla simple	"cadena1", 'cadena 2'
entero	Secuencia de uno o más dígitos	2007
nombre	Palabra Reservada	nombre
artista	Palabra Reservada	artista
ruta	Palabra Reservada	ruta
genero	Palabra Reservada	genero

repetir	Palabra Reservada	repetir
anio	Palabra Reservada	anio

2. Autómata grafico



Automata Finito del analisis lexico de un archivo con extensión (.mt3)

3. Expresiones Regulares de los tokens

token	E.R.
replis	replis
igual	=
flecha	=>
parentesis	(
parentesis)
dos puntos	:
punto y coma	;
coma	,
llave	{
llave	}
cadena	("." ('.')
entero	[0-9]+
nombre	nombre
artista	artista
genero	genero
ruta	ruta
repetir	repetir
anio	anio

4. Código fuente:

Método Analizar

Se encuentra ubicado en el archivo VentanaPrincipal.py donde llama al analizador léxico y conforme muestre los resultados de los análisis entonces devuelve el reporte que requiere:

```
def analizar():
    global contenido
    #print(contenido)
    a = AnalizadorLexico()
    if contenido != '':
        a.analizar(contenido)
        if len(a.listaErrores) == 0:
            ventana.destroy()
            generararchivoT(a.listaTokens)

            VentanaReproductor(nombreLista(a.listaTokens),aslistacan(aslistanombres(a.listaTokens),
aslistaartistas(a.listaTokens),aslistarutas(a.listaTokens),aslistageneros(a.listaTokens),
aslistarep(a.listaTokens),aslistaanios(a.listaTokens)))
        else:
            messagebox.showinfo("Warning","El archivo contiene errores, visualicelo en
el reporte")
            generararchivoE(a.listaErrores,a.listaTokens)
    else:
        messagebox.showinfo("Warning","No se puede analizar un archivo inexistente")
```

Función aslistacan:

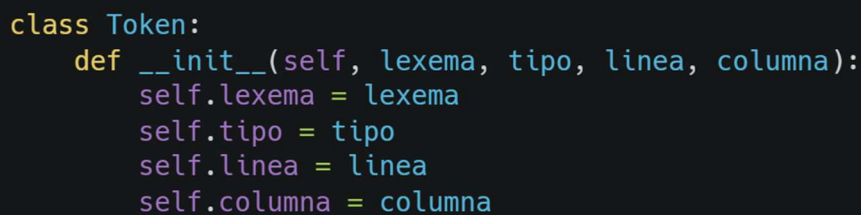
Esta función se encarga de que llame a todas las listas creadas de los atributos de la canción y se vuelva en una lista de objetos de canciones.

```
def aslistacan(listan,listaa,listar,listag,listare,listaan):
    listacanciones = []
    for i in range(len(listar)):
        listacanciones.append(Cancion(listan[i],listaa[i],listar[i],listag[i],listare[i],listaan[i]))
    return listacanciones
```

Para poder utilizar el analizador léxico fue útil el uso de estas dos clases:

Clase Token

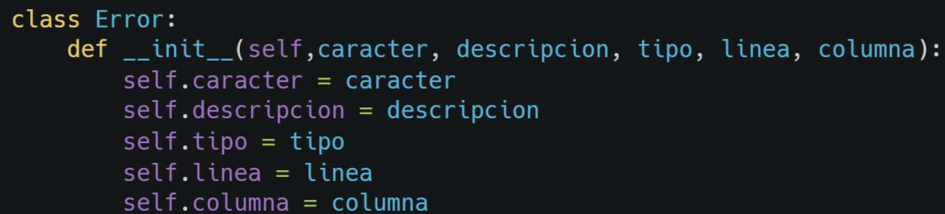
Esta clase hace que se guarden los objetos de tipo token aceptado.



```
class Token:
    def __init__(self, lexema, tipo, linea, columna):
        self.lexema = lexema
        self.tipo = tipo
        self.linea = linea
        self.columna = columna
```

Clase Error

Esta clase hace que se guarden los objetos donde haya un posible error léxico en su análisis.



```
class Error:
    def __init__(self, caracter, descripcion, tipo, linea, columna):
        self.caracter = caracter
        self.descripcion = descripcion
        self.tipo = tipo
        self.linea = linea
        self.columna = columna
```


Otra clase que fue útil pero en el desarrollo de la lógica del reproductor de música fue el uso de la clase Canción:

Clase Canción:

Esta clase se utiliza para guardar el contenido de las palabras reservadas que son atributos de un objeto canción.

```
class Cancion:
    def __init__(self,nombre,artista,ruta,genero,repeticion,anio):
        self.nombre = nombre
        self.artista = artista
        self.ruta = ruta
        self.genero = genero
        self.repeticion = repeticion
        self.anio = anio

    def getNombre(self):
        return self.nombre

    def getArtista(self):
        return self.artista

    def getRuta(self):
        return self.ruta

    def getGenero(self):
        return self.genero

    def getRepeticion(self):
        return self.repeticion

    def getAnio(self):
        return self.anio
        self.columna = columna
```

VI. Créditos

Elaborado por el alumno Rodrigo Alejandro Hernández de León para el curso de Lenguajes Formales y de Programación, en el país de Guatemala con fecha desde el 6 de diciembre de 2021 al 15 de diciembre de 2021.