MANUAL TÉCNICO

PROYECTO #1

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

FIUSAC
Universidad de San Carlos
de Guatemala

Lenguajes Formales y de Programación

Javier Alejandro Gutierrez de León

202004765

Guatemala, 23 de septiembre de 2021

Introducción

La empresa Bitxelart se dedica a la elaboración de imágenes digitales en estilo pixel art, para realizar estas piezas digitales un empleado toma una imagen que utilizará como referencia y la divide en una cuadrícula, anotando en un archivo de texto, con extensión pxla, las características de la misma (color, dimensiones, etc), luego este archivo es trasladado a un diseñador digital que utilizando el lenguaje de marcado HTML, dibuja la imagen solicitada.

En los últimos meses la demanda de estas piezas se ha multiplicado y al diseñador se le hace imposible terminar todos los trabajos encargados a tiempo, por lo que se le contrata a usted para que realice un programa que sea capaz de leer los archivos de texto de estas piezas y genere las imágenes correspondientes, para así automatizar este proceso y evitar pérdidas a la empresa.

La aplicación cuenta con un interfaz gráfica, en la cual se poseen las siguientes opciones:

1. **Cargar archivo:** Muestra una ventana emergente que permite al usuario seleccionar un archivo pxla y cargarlo a memoria.
2. **Analizar archivo:** Analiza el archivo cargado mediante un autómata definido e implementado por el estudiante y genera los archivos HTML tanto de la imagen original como de las imágenes con filtro. los datos obtenidos durante la ejecución del reporte.
3. **Ver reportes:** Se debe de escribir un archivo HTML con los datos del reporte generado, la manera en que los datos son mostrados deben de ser agradables al usuario. Los reportes deben abrirse automáticamente al seleccionar esta opción.
4. **Seleccionar imagen:** El programa debe contar con una lista de las imágenes analizadas en el archivo, se deberá poder seleccionar una imagen para poder visualizarla junto con los filtros aplicados en ella.
5. **Ver imagen:** El programa debe contar con un apartado para poder visualizar las imágenes generadas, para poder hacer esto se recomienda convertir el archivo HTML generado a un archivo con extensión jpg o png.
    a. **Original:** Muestra la imagen original sin aplicar ningún filtro.
    b. **MirrorX:** Muestra la imagen girada horizontalmente.
    c. **MirrorY:** Muestra la imagen girada verticalmente
    d. **DoubleMirror:** Muestra la imagen girada horizontal y verticalmente.

    Si se selecciona un filtro que no ha sido generado debe notificar al usuario.
6. **Salir:** Termina la ejecución de la aplicación.

Objetivos

Objetivos Generales

- ✔ Aplicar los conceptos adquiridos en el curso de Lenguajes Formales y de Programación en el desarrollo de la aplicación.

Objetivos Específicos

- ✔ Desarrolle una solución de software implementando un analizador léxico mediante autómatas.
- ✔ Aplique los conocimientos adquiridos en el laboratorio, del lenguaje de programación Python
- ✔ Desarrolle una interfaz gráfica utilizando el lenguaje Python

Especificación Técnica

Requisitos de Hardware

- **RAM: 128 MB**
- **Procesador: Mínimo Pentium 2 a 266 MHz**

Requisitos de software

- **Sistema operativo**
    - MS Windows XP o superior.
    - Apple OSX 10.4.x o superior.
    - GNU/Linux 2.6.x o superior.
- **Exploradores:**
    - **Google Chrome (en todas las plataformas)**
    - **Mozilla Firefox (en todas las plataformas)**
    - **Internet Explorer (Windows)**
    - **Microsoft Edge (Windows, Android, iOS, Windows 10 móvil)**
    - **Safari (Mac, iOS)**
    - **Opera (Mac, Windows)**

- **Lenguaje de Programación e IDE**

  Para el desarrollo de este software se utilizó Python, y como entorno se utilizó Visual Studio Code

- **Tecnologías utilizadas**
    - **Python (v 3.9.7):** Lenguaje de programación utilizado para desarrollar la aplicación.
    - **HTML5:** lenguaje de marcado para la elaboración de páginas web, con el que se realizaron los reportes generados desde Python.
    - **Tkinter:** Binding de la biblioteca gráfica Tcl/Tk para el lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario para Python y es el que viene por defecto con la instalación para Microsoft Windows.
    - **Boostrap:** Librería para el diseño del reporte HTML.
    - **Wkhtmltox:** Complemento para convertir html a imagen.
    - **html2image:** Librería para convertir de html a imagen.

Método del árbol realizado para generar el autómata finito determinista

Estructura del Archivo de entrada

```
TITULO="Pokebola";
ANCHO=300;
ALTO=300;
FILAS=12;
COLUMNAS=12;
CELDAS = {
[0,0,FALSE,#000000],
[0,1,FALSE,#000000],
[3,3,FALSE,#000000],
[3,4,TRUE,#000000],
[3,5,TRUE,#000000],
[3,6,TRUE,#000000],
[3,7,TRUE,#000000],
[4,1,FALSE,#000000]
};
FILTROS = MIRRORX;
@@@@
TITULO="Estrella";
ANCHO=300;
ALTO=300;
FILAS=4;
COLUMNAS=4;
CELDAS = {
[0,0,FALSE,#000000],
[1,1,FALSE,#000000],
[3,3,FALSE,#000000],
[2,1,FALSE,#000000]
};
FILTROS =
MIRRORX,MIRRORY,DOUBLEMIRROR;
```

Expresión regular

((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)*

PASO 1

Concatenar un símbolo de finalización a la expresión regular.

(((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)*)#

PASO 2

Construir el árbol binario

((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)* #

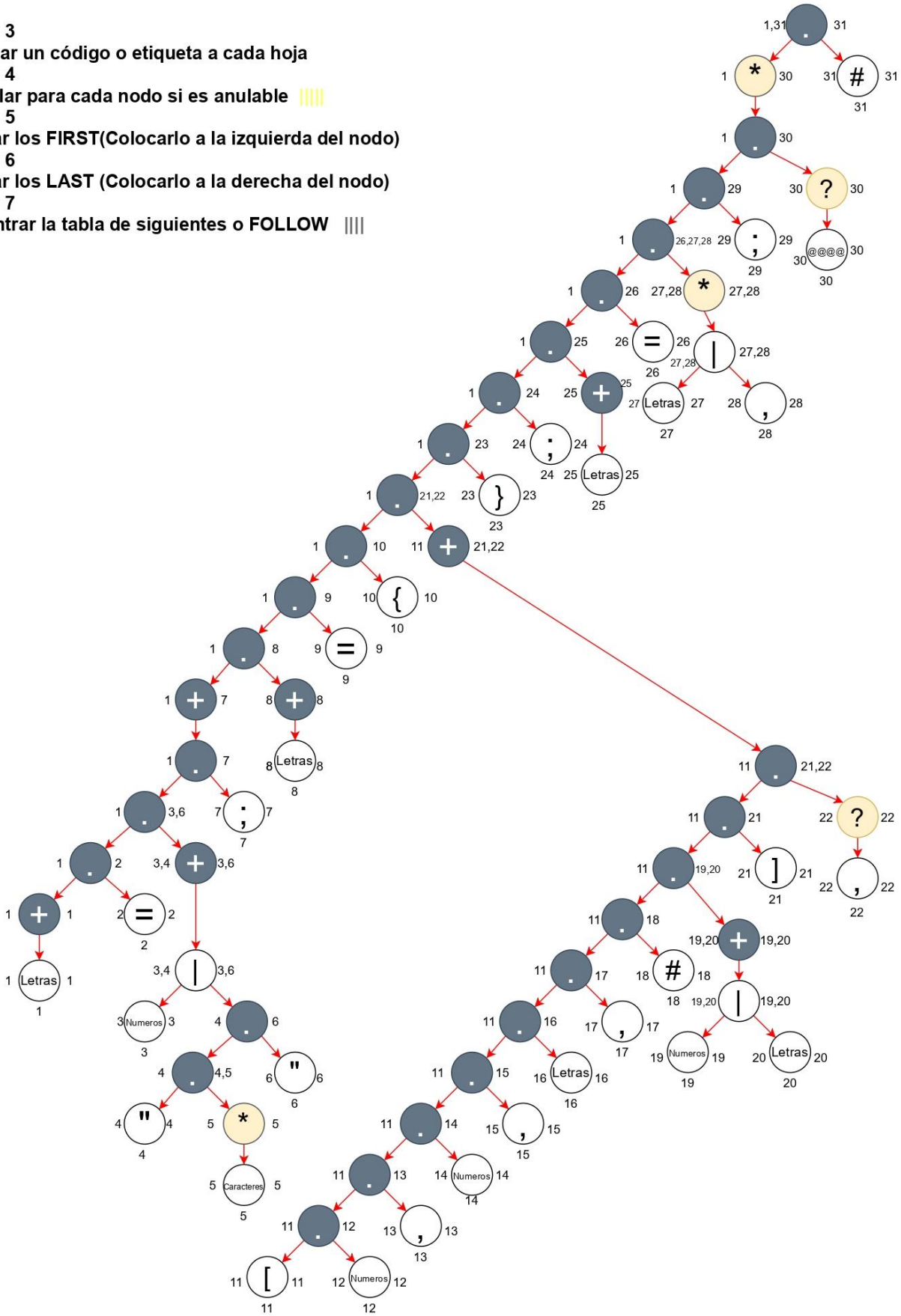
((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)* #

((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)* #

((Letras+ = ((" Caracteres* ")| Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# (Numeros|Letras)+)] ,?)+ } ; (Letras+ = (Letras | ,)* ;) (@ @ @ @)?)* #

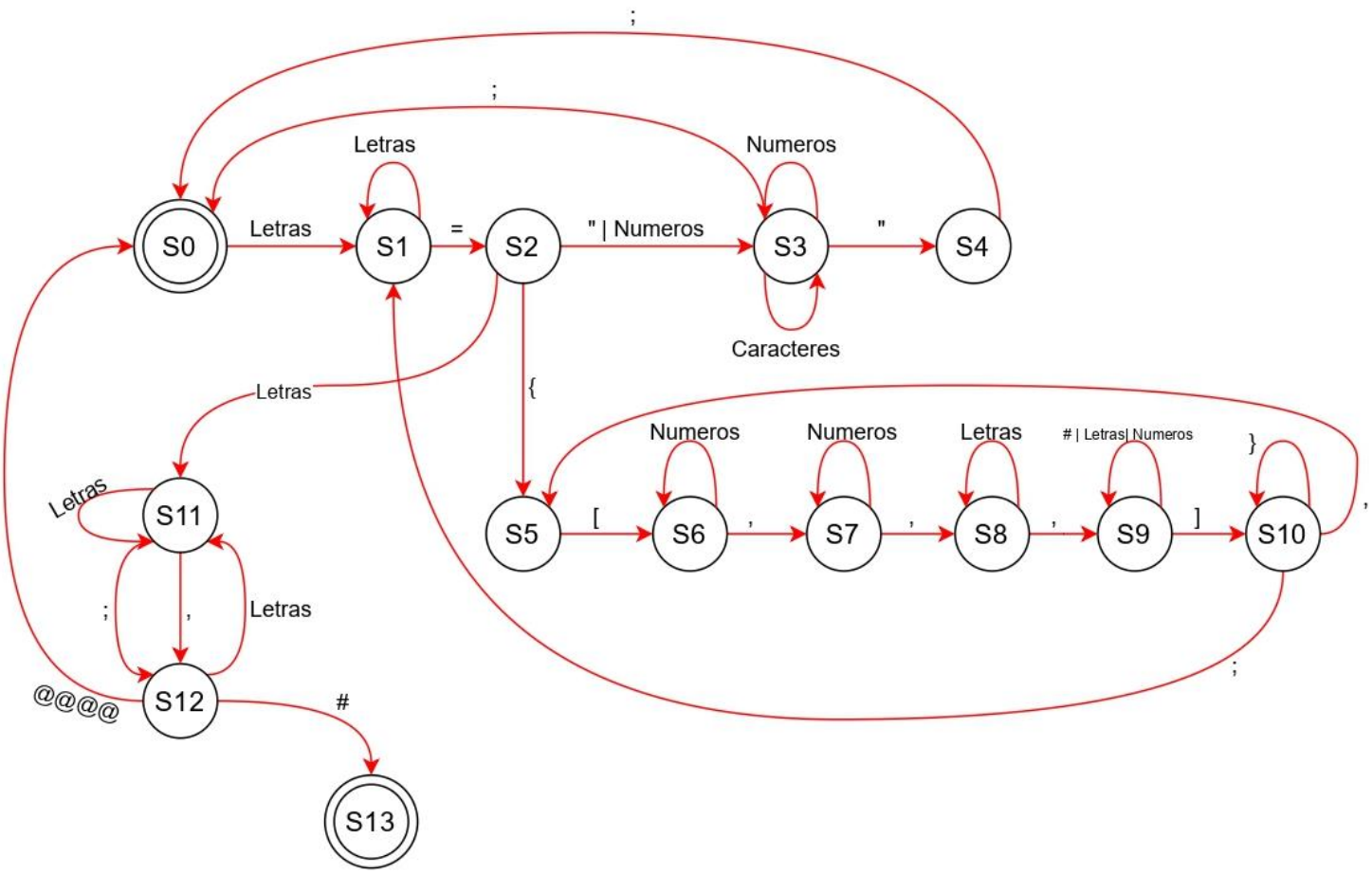
((Letras+ = (" Caracteres* ") | Numeros)+ ;)+ Letras+ = { ([Numeros , Numeros , Letras , (# Numeros+] ,)+ } ; (Letras+ = (Letras | ,)+ ;) (@ @ @ @)?)*

PASO 3
Colocar un código o etiqueta a cada hoja
PASO 4
Calcular para cada nodo si es anulable |||||
PASO 5
Indicar los FIRST(Colocarlo a la izquierda del nodo)
PASO 6
Indicar los LAST (Colocarlo a la derecha del nodo)
PASO 7
Encontrar la tabla de siguientes o FOLLOW ||||



Hojas	Terminal	Siguiente
1	Letras	1,2
2	=	3,4
3	Numeros	3,6,7
4	"	5,6
5	Caracteres	5,6
6	"	7
7	;	1,8
8	Letras	8,9
9	=	10
10	{	11
11	[	12
12	Numeros	12,13
13	,	14
14	Numeros	14,15
15	,	16
16	Letras	16,17
17	,	18
18	#	19,20
19	Numeros	19,20,21
20	Letras	19,20,21
21	]	22,23
22	,	23
23	}	24
24	;	25
25	Letras	25,26
26	=	27,28
27	Letras	27,28,29
28	,	27,28,29
29	;	30,31
30	@@@@	1
31	#	--

ESTADOS	Letras	=	Numeros	"	Caracteres	;	{	[	]	}	,	#	@@@@
S0={1}	S1={1,2}	--	--	--	--	--	--	--	--	--	--	--	--
S1={1,2}	S1={1,2}	S2={3,4}	--	--	--	--	--	--	--	--	--	--	--
S2={3,4}	--	--	S3={3,6,7}	S4={5,6}	--	--	--	--	--	--	--	--	--
S3={3,7}	--	--	S3={3,6,7}	--	--	S5={8}	--	--	--	--	--	--	--
S4={5,6}	--	--	--	--	S6={5,6}	--	--	--	--	--	--	--	--
S5={8}	S7={8,9}	--	--	--	--	--	--	--	--	--	--	--	--
S6={5,6}	--	--	--	S8={7}	S6={5,6}	--	--	--	--	--	--	--	--
S7={8,9}	S7={8,9}	S10={10}	--	--	--	--	--	--	--	--	--	--	--
S8={7}	--	--	--	--	--	S9={1,8}	--	--	--	--	--	--	--
S9={1,8}	S7={8,9}	--	--	--	--	--	--	--	--	--	--	--	--
S10={10}	--	--	--	--	--	--	S11={11}	--	--	--	--	--	--
S11={11}	--	--	--	--	--	--	--	S12={12}	--	--	--	--	--
S12={12}	--	--	S13={12,13}	--	--	--	--	--	--	--	--	--	--
S13={12,13}	--	--	S13={12,13}	--	--	--	--	--	--	--	S14={14}	--	--
S14={14}	--	--	S15={14,15}	--	--	--	--	--	--	--	--	--	--
S15={14,15}	--	--	S15={14,15}	--	--	--	--	--	--	--	S16={16}	--	--
S16={16}	S17={16,17}	--	--	--	--	--	--	--	--	--	--	--	--
S17={16,17}	S17={16,17}	--	--	--	--	--	--	--	--	--	S18={18}	--	--
S18={18}	--	--	--	--	--	--	--	--	--	--	--	S19={19,20}	--
S19={19,20}	S20={19,20,21}	--	S20={19,20,21}	--	--	--	--	--	--	--	--	--	--
S20={19,20,21}	S20={19,20,21}	--	S20={19,20,21}	--	--	--	--	--	S21={22,23}	--	--	--	--
S21={22,23}	--	--	--	--	--	--	--	--	--	S23={24}	S22={23}	--	--
S22={23}	--	--	--	--	--	--	--	--	--	S23={24}	--	--	--
S23={24}	S24={25}	--	--	--	--	--	--	--	--	--	--	--	--
S24={25}	S25={25,26}	--	--	--	--	--	--	--	--	--	--	--	--
S25={25,26}	S25={25,26}	S26={27,28}	--	--	--	--	--	--	--	--	--	--	--
S26={27,28}	S27={27,28,29}	--	--	--	--	--	--	--	--	--	S27={27,28,29}	--	--
S27={27,28,29}	S27={27,28,29}	--	--	--	--	S28={30,31}	--	--	--	--	S27={27,28,29}	--	--
S28={30,31}	--	--	--	--	--	S29={FIN}	--	--	--	--	--	--	S1={1,2}
S29={FIN}	--	--	--	--	--	--	--	--	--	--	--	--	--


;
;
Letras
Numeros
S0
Letras
S1
=
S2
" | Numeros
S3
"
S4
Caracteres
Letras
{
Numeros
Numeros
Letras
# | Letras| Numeros
}
Letras
S11
S5
[
S6
,
S7
,
S8
,
S9
]
S10
,
;
,
Letras
@@@@
S12
#
S13

Lógica del programa (Métodos principales)

Nombre	Descripción	Parámetros
App()	Clase Principal, en la cual se creaba toda la interfaz gráfica y se llamaban a las funciones para las opciones del programa.	• **Ninguno**
act(self,LTitulo,img0,img1,img2,img3)	Función para actualizar la ruta de las imágenes que se debian mostrar en el apartado de "Ver Imagen" incluyendo los filtros.	• **LTitulo:** Label en el cual se actualiza el nombre de la imagen. • **img0:** Ruta para encontrar la imagen original. • **img1:** Ruta para encontrar la imagen con el filtro de MIRRORX. • **img2:** Ruta para encontrar la imagen con el filtro de MIRRORY. • **img3:** Ruta para encontrar la imagen con el filtro de DOUBLEMIRROR.
LeerArchivo(Ruta,La1,Ruta2)	Función para buscar y guardar en memoria la ruta del archivo que se desea analizar mas adelante.	• **Ruta:** Label que muestra la ruta seleccionada en el Frame de "Cargar Archivo" • **La1:** Label del titulo si se encontro la imagen. • **Ruta2:** Label que muestra la ruta seleccionada en el Frame de "Analizar Archivo"
leerArchivo(ruta)	Función con la que se abre el archivo mediante la ruta previamente seleccionada.	• **Ruta:** Ruta seleccionada en el Frame de "Cargar Archivo"
AnalizarArchivo(ruta,myframe2, lt,ltt,img0,img1,img2,img3)	Función con la que se analiza el archivo que en memoria y se generan reportes, imágenes y las listas dentro de la interfaz del programa por medio de otras funciones.	• **Ruta:** Ruta seleccionada en el Frame de "Cargar Archivo" • **myframe2:** Frame donde se actualizará la lista de imágenes. • **lt:** Label del titulo de la imagen seleccionada. • **ltt:** Label del titulo2 de la imagen seleccionada. • **img0:** Ruta para encontrar la imagen original. • **img1:** Ruta para encontrar la imagen con el filtro de MIRRORX. • **img2:** Ruta para encontrar la imagen con el filtro de MIRRORY. • **img3:** Ruta para encontrar la imagen con el filtro de DOUBLEMIRROR.

generarRTokens(lista)	Función para generar un reporte HTML de la lista de tokens aceptados por el Analizador Léxico.	• **Lista:** Lista de tokens aceptados.
generarRErrores(listaE)	Función para generar un reporte HTML de la lista de errores detectados por el Analizador Léxico.	• **ListaE:** Lista de errores identificados.
generarHTML(imagen)	Función para generar la imagen original utilizando una tabla de HTML, para luego convertirla a png.	• **Imagen:** Lista generada para almacenar el objeto Imagen.
generarHTML_MX(imagen)	Función para generar la imagen con el filtro MIRRORX utilizando una tabla de HTML, para luego convertirla a png.	• **Imagen:** Lista generada para almacenar el objeto Imagen.
generarHTML_MY(imagen)	Función para generar la imagen con el filtro MIRRORY utilizando una tabla de HTML, para luego convertirla a png.	• **Imagen:** Lista generada para almacenar el objeto Imagen.
generarHTML_MXY(imagen)	Función para generar la imagen con el filtro DOUBLEMIRROR utilizando una tabla de HTML, para luego convertirla a png.	• **Imagen:** Lista generada para almacenar el objeto Imagen.
LlenarT(myframe2,lt,ltt,img0, img1,img2,img3)	Función para llenar las listas de radioButtons con las imágenes que se encuentran en el archivo.	• **myframe2:** Frame donde se actualizarán lasimágenes. • **lt:** Label del titulo de la imagen seleccionada. • **ltt:** Label del titulo2 de la imagen seleccionada. • **img0:** Ruta para encontrar la imagen original. • **img1:** Ruta para encontrar la imagen con el filtro de MIRRORX. • **img2:** Ruta para encontrar la imagen con el filtro de MIRRORY. • **img3:** Ruta para encontrar la imagen con el filtro de DOUBLEMIRROR.
Original(tabimg,img0)	Funcion para buscar y mostrar la imagen original.	• **myframe2:** Frame donde se actualizarán lasimágenes. • **Img0:** Ruta para encontrar la imagen original.
MX(tabimg,img1)	Funcion para buscar y mostrar la imagen con el filtro MIRRORX.	• **myframe2:** Frame donde se actualizarán lasimágenes. • **Img1:** Ruta para encontrar la imagen con el filtro de MIRRORX.

MY(tabimg,img2)	Funcion para buscar y mostrar la imagen con el filtro MIRRORY.	• **myframe2:** Frame donde se actualizarán lasimágenes. • **Img2:** Ruta para encontrar la imagen con el filtro de MIRRORY.
MXY(tabimg,img3)	Funcion para buscar y mostrar la imagen con el filtro DOUBLEMIRROR.	• **myframe2:** Frame donde se actualizarán lasimágenes. • **Img3:** Ruta para encontrar la imagen con el filtro de DOUBLEMIRROR.
Imagen(self,titulo,ancho,alto, filas, columnas,celdas,filtros)	Función para guardar los atrbutos de cada imagen que se encuentra en el archivo de entrada.	• **titulo:** Atributo titulo de la imagen. • **ancho:** Atributo ancho de la imagen. • **alto:** Atributo alto de la imagen. • **filas:** Atributo filas de la imagen. • **columnas:** Atributo columnas de la imagen. • **celdas:** Atributo celdas de la imagen. • **filtros:** Atributo filtros de la imagen.

Conclusiones

El uso del automata creado a travez del metodo del arbol ayuda a simplificar y optimizarlo de manera de que este se vuelva lo mas corto posible, a como se realizaría con otro tipo de métodos, y que a la hora de implementarlo la simplicidad que nos ofrece Python favoreció a la implementacion rapida y eficiente del autómata.
El uso de la librería Tkinter para desarrollar la interfaz gráfica favoreció a la facil creación de forma dinámica de los objetos que debían aparecer según los datos ingresados por medio del archivo.