MANUAL TÉCNICO

PROYECTO #1 LFP

Tabla de Contenido

Objetivos

- **Específicos**
  - Análisis texto mediante un lenguaje definido
- **Generales**
  - Generar formulario dinámico
  - Cargar archivos

Dirigido a cualquier usuario que desee generar formularios a partir de un archivo .form que tenga la estructura del lenguaje definido.

Especificación Técnica

Requisitos de Hardware

- Mouse
- Teclado

Requisitos de Software

- **Sistema Operativo:** Windows 10, Linux, MAC OS
- **Herramientas:** Visual Studio Code
- **Lenguaje de Programación:** Python

Expresión Regular

~|>|[|<|:|"|,|'|]|-|[0-9]+|[A-Za-z0-9]*

Autómata

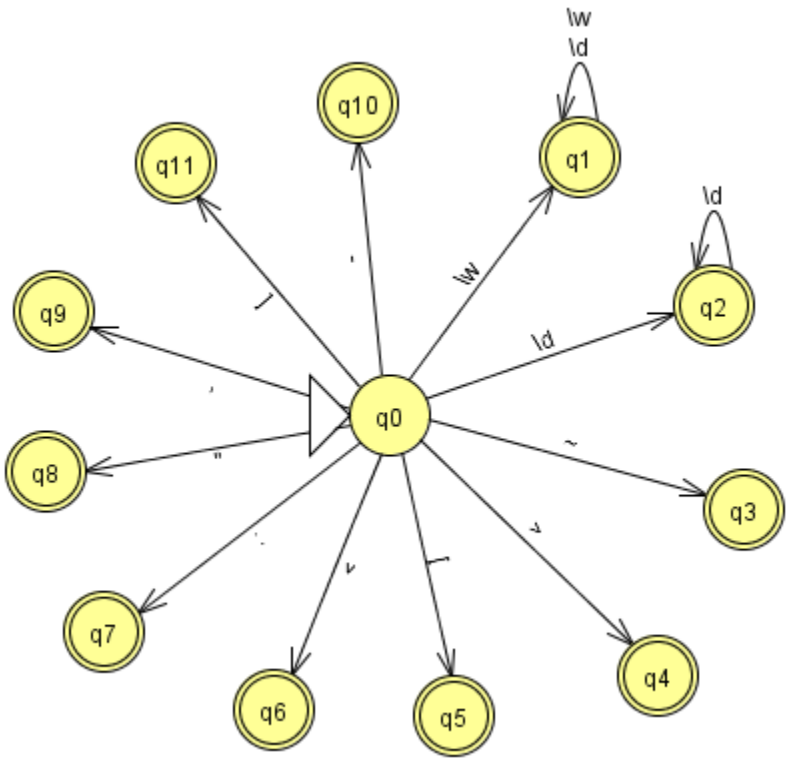


Tabla de Tokens

Descripción	Patrón	Expresión Regular	Ejemplos
Palabra Reservada	Palabra	[A-Za-z]	formulario
Signo raro	Un caracter '~'	'~'	~
Identificadores	Una letra seguida de una o más letras	[A-Za-z]	tipo, valor, fondo, evento
Corchete Izquierdo	Un caracter '['	'['	[
Signo menor que	Un caracter '<'	'<'	<
Dos puntos	Un caracter ':'	':'	:
Comilla doble	Un caracter ' " '	' " '	"
Coma	Un caracter ','	','	,
Signo mayor que	Un caracter '>'	'>'	>
Comilla simple	Un caracter " ' "	" ' "	'
Corchete derecho	Un caracter ']'	']'	]
Signo menos	Un caracter '-'	'-'	-
Parametros del HTML	Palabra	[A-Za-z]	etiqueta, texto, Nombre:, sexo, Ingrese nombre

Lógica del Programa

Clases Utilizadas:

- **Main**
- **Analizador**
- **Gestor**
- **Error**
- **Token**
- **Contenido**
- **Interfaz**

➢ **Main**

Clase principal, la cual se tiene que ejecutar para utilizar el programa.

```python
from Interfaz import Interfaz


if __name__ == '__main__':
    app = Interfaz()
```

➢ **Analizador**

Clase donde se establecen todos los métodos y funciones que se utilizan para codificar el autómata que permite la lectura del archivo.

```python
class Analizador:

    def __init__(self):
        self.Tokens = []
        self.Errores = []
        self.linea = 1
        self.columna = 0
        self.buffer = ''
        self.estado = 0
        self.i = 0
        self.listaTokens=[]
        self.listaErrores=[]

    def agregarToken(self,caracter, linea, columna, token):
        self.Tokens.append(Token(caracter,linea,columna,token))
        self.buffer = ''

    def agregarError(self,caracter, linea, columna):
        self.Errores.append(Error('Caracter ' + caracter + ' no reconocido.', linea, columna))
```

- **Gestor**

Clase donde se establece la ventana emergente que se abrirá para seleccionar el archivo .form

```
class Gestor:

    def __init__(self):
        pass

    def rutaArchivo(self):
        ruta = easygui.fileopenbox()
        return ruta

    def CargarData(self):
        data = self.rutaArchivo()
        archivo = open(data, 'r', encoding = "utf-8")
        texto = archivo.read()
        #SIMBOLO TERMINAL
        texto+='\n$'
        archivo.close()
        return texto
```

- **Error**

Clase donde se almacenarán los objetos para los errores encontrados en el análisis.

```
class Error:

    def __init__(self, descripcion : str, linea : int, columna : int):
        self.descripcion = descripcion
        self.linea = linea
        self.columna = columna

    def imprimirError(self):
        print(self.descripcion, self.linea, self.columna)
```

- **Token**

Clase donde se almacenarán los objetos para los tokens encontrados en el análisis.

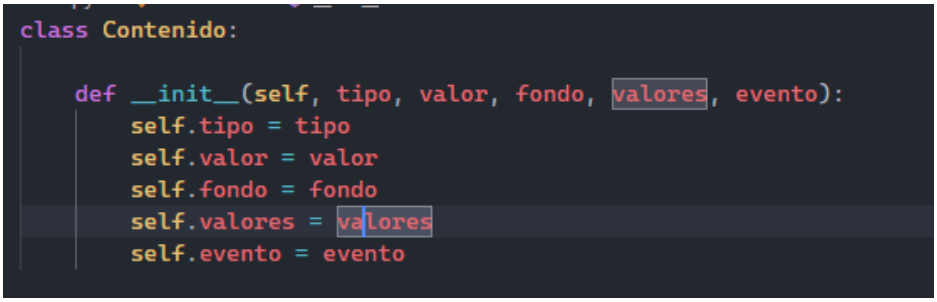
```
class Token:

    def __init__(self, lexema : str, linea : int, columna : str, tipo : str):
        self.lexema = lexema
        self.linea = linea
        self.columna = columna
        self.tipo = tipo

    def imprimirToken(self):
        print(self.lexema, self.linea, self.columna, self.tipo)
```

- **Contenido**

Clase utilizada para el método de lectura de los tokens, que se utilizara para generar el formulario.



```
class Contenido:

    def __init__(self, tipo, valor, fondo, valores, evento):
        self.tipo = tipo
        self.valor = valor
        self.fondo = fondo
        self.valores = valores
        self.evento = evento
```

- **Interfaz**

Clase que genera la interfaz del programa.

```
15
16  class Interfaz:
17      def __init__(self):
18          self.evento = []
19          self.Texto = []
20          self.Etiqueta = []
21          self.Boton = []
22          self.Contenido = []
23
24          raiz = Tk()
25          raiz.title("Analizador, Proyecto1 LFP")
26          raiz.resizable(0,0)
27          raiz.iconbitmap("Icono.ico")
28          #raiz.geometry("850x550")
29          #raiz.config(bg="White")
30          miFrame = Frame()
31          miFrame.pack()
32          #miFrame.config(bg="White")
33          miFrame.config(width="850",height="550")
34
35          #self.texto = StringVar()
```