



MANUAL TÉCNICO

PROYECTO 2

ALDO SAÚL VÁSQUEZ MOREIRA
CARNET 202109754
LAB. LENGUAJES FORMALES Y DE PROG.

ENCABEZADO

Nombre: Aldo Saúl Vásquez Moreira

Carnet: 20109754

Laboratorio Lenguajes Formales y de Programación.

Nombre de Sistema: Generador de páginas web.

PRINCIPIO, TÉCNICA O PARADIGMA APLICADO DE PROGRAMACIÓN

Se utilizó el paradigma de Programación Orientado a Objetos con Python.

CONVENCIONES DE NOMENCLATURA

- Se declararon las clases con letra inicial mayúscula.
- Se declararon los métodos con letra inicial minúscula y aplicando la convención Snake Case y Camel Case.
- Se declararon las variables con letra inicial minúscula y con un guion bajo en caso fueran necesarias más de dos palabras.

DIAGRAMA DE CLASES

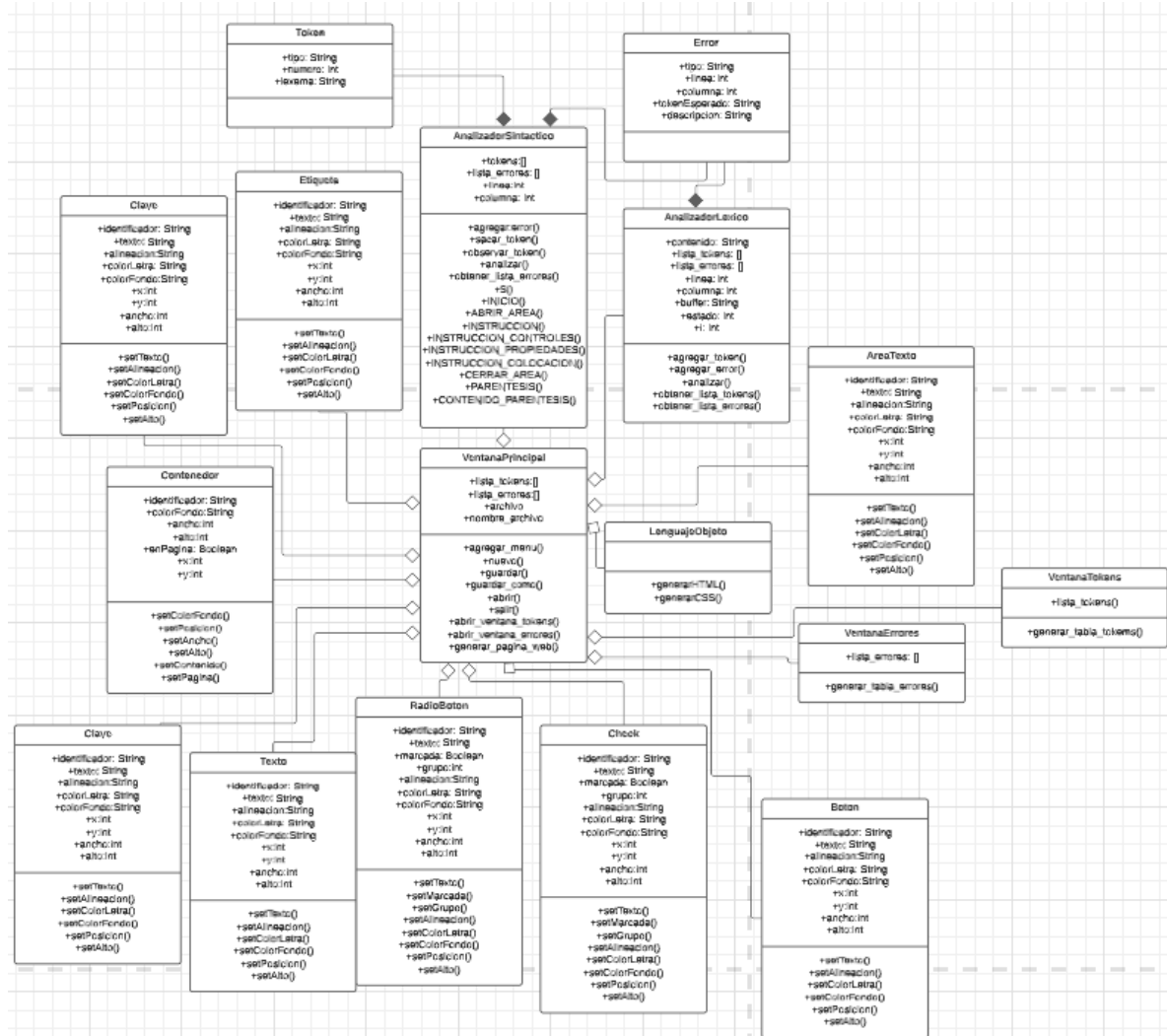


TABLA DE TOKENS

TOKEN	EXPRESIÓN REGULAR
Reservada	L+#1
Control	L(L D)*#2
ID	L(L D)*#3
Punto y coma	;;#4
Propiedad	L+#5
Valor	(L+ D+)#6
Punto	.#7
Coma	,#8
Paréntesis izquierdo	(#9
Paréntesis derecho)#10
Mayor que	>#11
Menor que	<#12
Guion	-#13
Exclamación	!#14
Comillas	"#15

METODO DEL ÁRBOL UTILIZADO

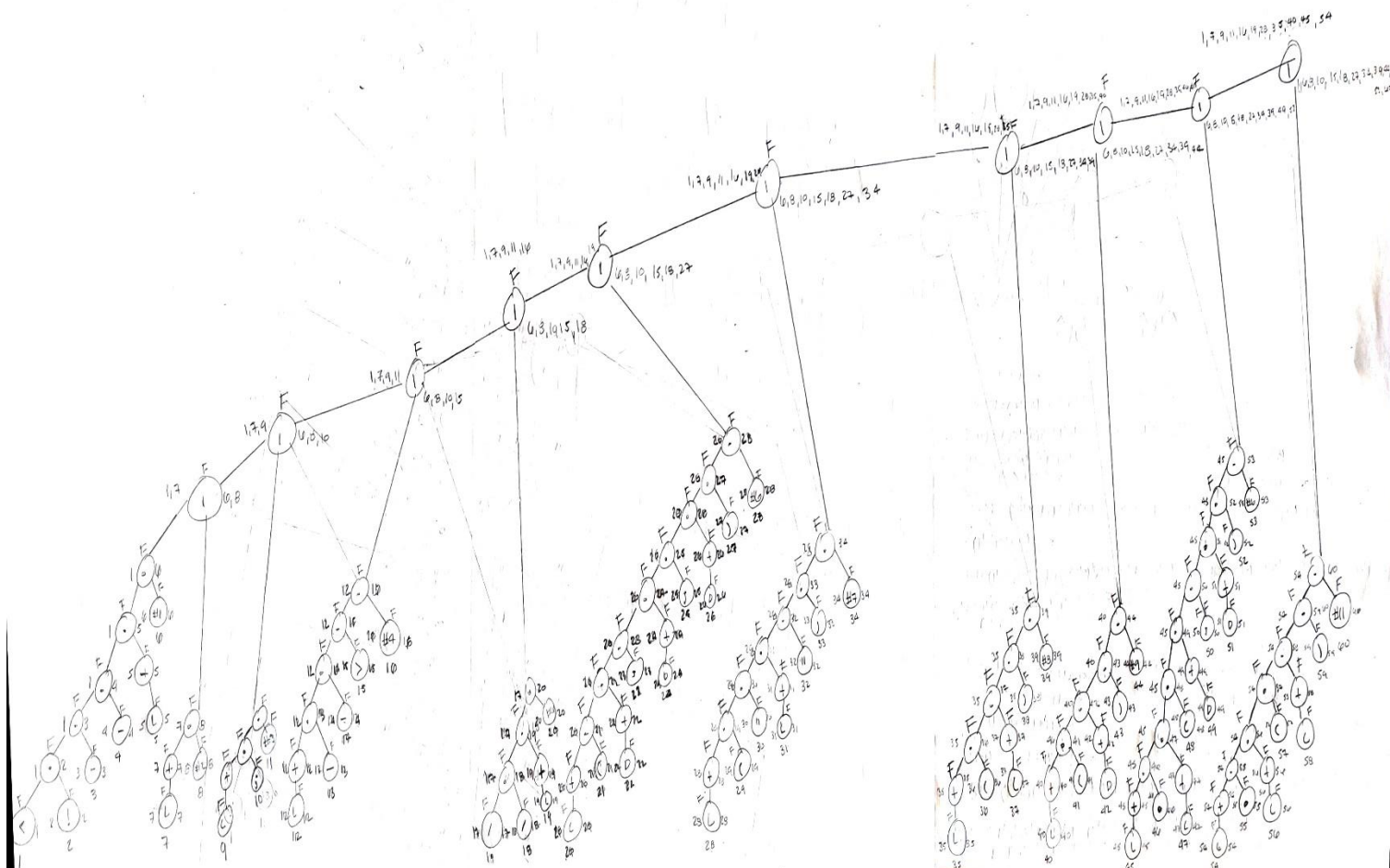
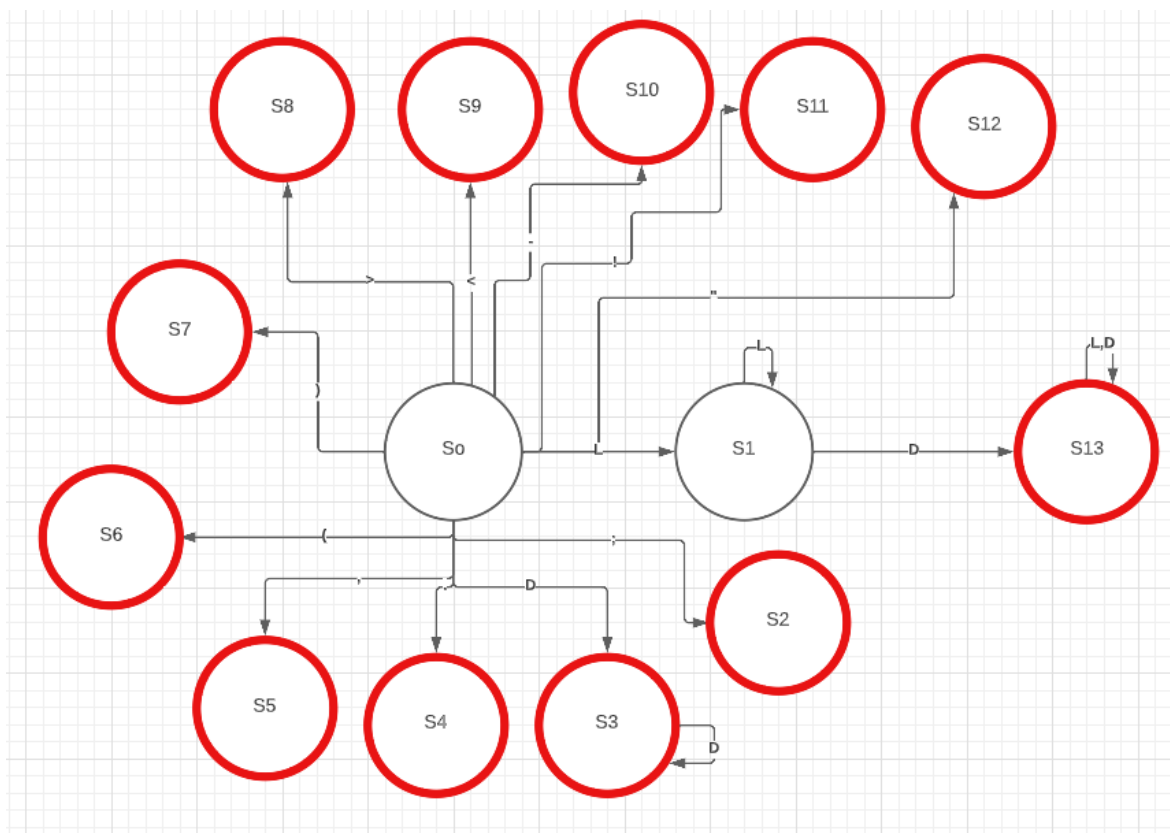


TABLA DE PRIMEROS, ULTIMOS Y SIGUIENTES

i	sig(i)
1 - L	1, 2
2 - #1	
3 - L	4, 5, 6
4 - L	4, 5, 6
5 - D	4, 5, 6
6 - #2	
7 - L	8, 9, 10
8 - L	8, 9, 10
9 - D	8, 9, 10
10 - #3	
11 - ;	12
12 - #4	
13 - L	13, 14
14 - #5	
15 - L	15, 17
16 - D	16, 17
17 - #6	
18 - .	19
19 - #7	
20 - ,	21
21 - #8	
22 - (23
23 - #9	
24 -)	25
25 - #10	
26 - >	27
27 - #11	
28 - <	29
29 - #12	
30 - -	31
31 - #13	
32 - !	33
33 - #14	
34 - "	35
35 - #15	

GRÁFICO DE AUTÓMATA FINITO DETERMINISTA



GRAMÁTICA LIBRE DE CONTEXTO PARA EL ANÁLISIS SINTÁCTICO

$S \rightarrow \text{INICIO}$

$\text{INICIO} \rightarrow \text{ABRIRAREA}$

$\text{ABRIRAREA} \rightarrow \text{INSTRUCCION}$

$\text{INSTRUCCION} \rightarrow \text{INSTRUCCIONCONTROL INSTRUCCION}$

$\quad | \text{INSTRUCCIOPROPIEDADES INSTRUCCION}$

$\quad | \text{INSTRUCCIONCOLOCACION INSTRUCCION}$

$\quad | \text{CERRARAREA}$

$\text{CERRARAREA} \rightarrow \text{INICIO}$

MÉTODOS PRINCIPALES

- **generar_pagina_web:** Este método se encarga de enlazar el analizador léxico y sintáctico, así como el generador del lenguaje objeto para generar la página web.

```
104     def generar_pagina_web(self):
105         self.lista_tokens.clear()
106         self.lista_errores.clear()
107         #obteniendo contenido del archivo abierto
108         self.archivo = open(self.nombre_archivo, 'r', encoding='utf-8')
109         contenido = self.archivo.read()
110         self.archivo.close()
111         #realizando analisis lexico
112         analizador_lexico = AnalizadorLexico(contenido)
113         analizador_lexico.analizar()
114         #obteniendo lista de tokens del analisis lexico
115         self.lista_tokens += analizador_lexico.obtener_lista_tokens()
116         #obteniendo lista de errores del analisis léxico
117         self.lista_errores += analizador_lexico.obtener_lista_errores()
118
119         #lista de tokens secundaria para enviar al analizador sintactico
120         tokens = []
121         for i in range (len(self.lista_tokens)):
122             tokens.append(self.lista_tokens[i])
123
124         #creando analizador sintáctico y enviandole la lista de tokens
125         analizador_sintactico = AnalizadorSintactico(tokens)
126
127         #creando analizador sintáctico y enviandole la lista de tokens
128         analizador_sintactico = AnalizadorSintactico(tokens)
129         analizador_sintactico.analizar()
130         self.lista_errores += analizador_sintactico.obtener_lista_errores()
131
132         if len(self.lista_errores) == 0:
133             tk.messagebox.showinfo(message="Archivo compilado correctamente.", title="Éxito")
134             #creando listas de los diferentes controles en el archivo de entrada
135             lista_etiquetas = []
136             lista_botones = []
137             lista_checks = []
138             lista_radioBotones = []
139             lista_textos = []
140             lista_areasTexto = []
141             lista_claves = []
142             lista_contenedores = []
143
144             #analizando la lista de tokens para identificar sus propiedades y posicion para html y css
145             for i in range(len(self.lista_tokens)):
146                 token = self.lista_tokens[i]
147                 #contenedores
148                 if token.numero == 3 and (token.lexema == 'Contenedor' or token.lexema == 'contenedor'):
```

- **generar_tabla_tokens:** Este método se encarga de generar la tabla de tokens cuando se analiza un archivo.


```
def generar_tabla_tokens(self):
    #creando encabezado de tabla
    tabla = ttk.Treeview(self.ventana, columns=('0', '1'))
    tabla.grid(row=1, column=0, columnspan=2)
    tabla.heading('#0', text = 'Número')
    tabla.heading('#1', text = 'Tipo')
    tabla.heading('#2', text = 'Lexema')

    #llenando la tabla
    for token in self.lista_tokens:
        tabla.insert('', 0, text=token.numero, values=(token.tipo, token.lexema))
```

- **generar_tabla_errores:** Este método se encarga de generar la tabla de errores cuando se analiza un archivo.

```
def generar_tabla_errores(self):
    #creando encabezado de tabla
    tabla = ttk.Treeview(self.ventana, columns=('0', '1', '2', '3'))
    tabla.grid(row=1, column=0, columnspan=2)
    tabla.heading('#0', text = 'Tipo')
    tabla.heading('#1', text = 'Línea')
    tabla.heading('#2', text = 'Columna')
    tabla.heading('#3', text = 'Componente esperado')
    tabla.heading('#4', text = 'Descripción')

    #llenando la tabla
    for error in self.lista_errores:
        tabla.insert('', 0, text=error.tipo, values=(error.linea, error.columna, error.tokenEsperado, error.de
```

- **generarHTML:** Este método se encarga de generar el código HTML para dar la estructura de la página web.

```
def generarHTML(self, contenedores, etiquetas, botones, textos, areasTexto, claves, checks, radios):
    with open('../Salida/index.html', 'w') as archivo:
        archivo.write('<html>\n')
        archivo.write('<head>\n')
        archivo.write('<link href="prueba.css" rel="stylesheet" type="text/css"/>\n')
        archivo.write('</head>\n')
        archivo.write('<body>\n')

        for contenedor in contenedores:
            #agregando al html los div externos
            if contenedor.enPagina == True:
                archivo.write(f'    <div id="{contenedor.identificador}">\n')
                if len(contenedor.contieneA) > 0:
                    for contenido in contenedor.contieneA:
                        archivo.write(f'        <div id="{contenido}">\n')
                        archivo.write('        </div>\n')
                    archivo.write('    </div>\n')
                archivo.write('    </div>\n')

        for etiqueta in etiquetas:
            archivo.write(f'    <label id="{etiqueta.identificador}">{etiqueta.texto}</label>\n')
```

```

28         for etiqueta in etiquetas:
29             archivo.write(f'    <label id="{etiqueta.identificador}">{etiqueta.texto}</label>\n')
30
31         for boton in botones:
32             archivo.write(f'    <input type="submit" id="{boton.identificador}" value="{boton.texto}" style="t
33
34         for texto in textos:
35             archivo.write(f'    <input type="text" id="{texto.identificador}" value="{texto.texto}" style="tex
36
37         for areaTexto in areasTexto:
38             archivo.write(f'    <TEXTAREA id="{areaTexto.identificador}">{areaTexto.texto}</TEXTAREA>\n')
39
40         for clave in claves:
41             archivo.write(f'    <input type="password" id="{clave.identificador}" value="{clave.texto}" style=
42
43         for check in checks:
44             archivo.write(f'    <input type="checkbox" id="{check.identificador}" {check.marcada}/>{check.text
45
46         for radio in radios:
47             archivo.write(f'    <input type="radio" name="{radio.grupo}" id="{radio.identificador}" {radio.mar
48
49         archivo.write('</body>\n')

```

- **generarCSS:** Este método se encarga de generar el archivo CSS para dar estilos a la página web.

```

51
52     def generarCSS(self, contenedores, etiquetas, botones, textos, areasTexto, claves, checks, radios):
53         with open('../Salida/prueba.css', 'w') as archivo:
54
55             for etiqueta in etiquetas:
56                 archivo.write(f'#{etiqueta.identificador}' + '\n')
57                 archivo.write('position:absolute;\n')
58                 archivo.write(f'top:{etiqueta.y}px;\n')
59                 archivo.write(f'left:{etiqueta.x}px;\n')
60                 archivo.write(f'width:{etiqueta.ancha}px;\n')
61                 archivo.write(f'height:{etiqueta.alto}px;\n')
62                 archivo.write(f'color:rgb({etiqueta.colorLetra});\n')
63                 archivo.write(f'background-color:rgb({etiqueta.colorFondo});\n')
64                 archivo.write(f'font-size:12px;\n')
65                 archivo.write('}\n')
66
67             for contenedor in contenedores:
68                 archivo.write(f'#{contenedor.identificador}' + '\n')
69                 archivo.write('position:absolute;\n')
70                 archivo.write(f'top:{contenedor.y}px;\n')
71                 archivo.write(f'left:{contenedor.x}px;\n')
72                 archivo.write(f'width:{contenedor.ancha}px;\n')

```

```

77
78     for texto in textos:
79         archivo.write(f'#{texto.identificador}' + '{\n')
80         archivo.write('position:absolute;\n')
81         archivo.write(f'top:{texto.y}px;\n')
82         archivo.write(f'left:{texto.x}px;\n')
83         archivo.write(f'width:{texto.anch}px;\n')
84         archivo.write(f'height:{texto.alto}px;\n')
85         archivo.write(f'color:rgb({texto.colorLetra});\n')
86         archivo.write(f'background-color:rgb({texto.colorFondo});\n')
87         archivo.write(f'font-size:12px;\n')
88         archivo.write('}\n')
89
90     for clave in claves:
91         archivo.write(f'#{clave.identificador}' + '{\n')
92         archivo.write('position:absolute;\n')
93         archivo.write(f'top:{clave.y}px;\n')
94         archivo.write(f'left:{clave.x}px;\n')
95         archivo.write(f'width:{clave.anch}px;\n')
96         archivo.write(f'height:{clave.alto}px;\n')
97         archivo.write(f'color:rgb({clave.colorLetra});\n')
98         archivo.write(f'background-color:rgb({clave.colorFondo});\n')
99         archivo.write(f'font-size:12px;\n')
100        archivo.write('}\n')
101
102    for boton in botones:
103        archivo.write(f'#{boton.identificador}' + '{\n')
104        archivo.write('position:absolute;\n')
105        archivo.write(f'top:{boton.y}px;\n')
106        archivo.write(f'left:{boton.x}px;\n')

```

```

114    for areaTexto in areasTexto:
115        archivo.write(f'#{areaTexto.identificador}' + '{\n')
116        archivo.write('position:absolute;\n')
117        archivo.write(f'top:{areaTexto.y}px;\n')
118        archivo.write(f'left:{areaTexto.x}px;\n')
119        archivo.write(f'width:{areaTexto.anch}px;\n')
120        archivo.write(f'height:{areaTexto.alto}px;\n')
121        archivo.write(f'color:rgb({areaTexto.colorLetra});\n')
122        archivo.write(f'background-color:rgb({areaTexto.colorFondo});\n')
123        archivo.write(f'font-size:12px;\n')
124        archivo.write('}\n')
125
126    for check in checks:
127        archivo.write(f'#{check.identificador}' + '{\n')
128        archivo.write('position:absolute;\n')
129        archivo.write(f'top:{check.y}px;\n')
130        archivo.write(f'left:{check.x}px;\n')
131        archivo.write(f'width:{check.anch}px;\n')
132        archivo.write(f'height:{check.alto}px;\n')
133        archivo.write(f'color:rgb({check.colorLetra});\n')
134        archivo.write(f'background-color:rgb({check.colorFondo});\n')
135        archivo.write(f'font-size:12px;\n')
136        archivo.write('}\n')
137
138    for radio in radios:
139        archivo.write(f'#{radio.identificador}' + '{\n')
140        archivo.write('position:absolute;\n')
141        archivo.write(f'top:{radio.y}px;\n')
142        archivo.write(f'left:{radio.x}px;\n')
143        archivo.write(f'width:{radio.anch}px;\n')

```

- **obtener_lista_tokens:** Este método retorna la lista de tokens que necesita el analizador sintáctico.

```

#retornando lista de tokens y errores
def obtener_lista_tokens(self):
    return self.lista_tokens

```

- **Obtener_lista_errores:** Este método retorna la lista de errores.

```
def obtener_lista_errores(self):
    return self.lista_errores
```

- **analizar:** Este método utiliza cada uno de los estados del autómata finito determinista para realizar el análisis léxico.

```
#analizando cadenas del archivo
def analizar(self):
    self.lista_errores = []
    self.lista_tokens = []
    lineas = self.contenido.split('\n')

    for linea in lineas:
        self.columna = 0
        #eliminando tabulaciones
        linea_tabulacion = linea.replace('\t', '')
        #eliminando espacios
        palabras = linea_tabulacion.split(' ')

        #recorriendo palabras de la linea
        for cadena in palabras:
            self.buffer = ''
            self.estado = 0
            self.i = 0
            cadena += '$'
            while self.i < len(cadena):
                if self.estado == 0:
                    self.s0(cadena[self.i])
                elif self.estado == 1:
                    self.s1(cadena[self.i])
                elif self.estado == 2:
                    self.s2(cadena[self.i])
                elif self.estado == 3:
```

```
                elif self.estado == 2:
                    self.s2(cadena[self.i])
                elif self.estado == 3:
                    self.s3(cadena[self.i])
                elif self.estado == 4:
                    self.s4(cadena[self.i])
                elif self.estado == 5:
                    self.s5(cadena[self.i])
                elif self.estado == 6:
                    self.s6(cadena[self.i])
                elif self.estado == 7:
                    self.s7(cadena[self.i])
                elif self.estado == 8:
                    self.s8(cadena[self.i])
                elif self.estado == 9:
                    self.s9(cadena[self.i])
                elif self.estado == 10:
                    self.s10(cadena[self.i])
                elif self.estado == 11:
                    self.s11(cadena[self.i])
                elif self.estado == 12:
                    self.s12(cadena[self.i])
                elif self.estado == 13:
                    self.s13(cadena[self.i])
                self.i += 1
            self.linea += 1
229
```

DESCRIPCIÓN

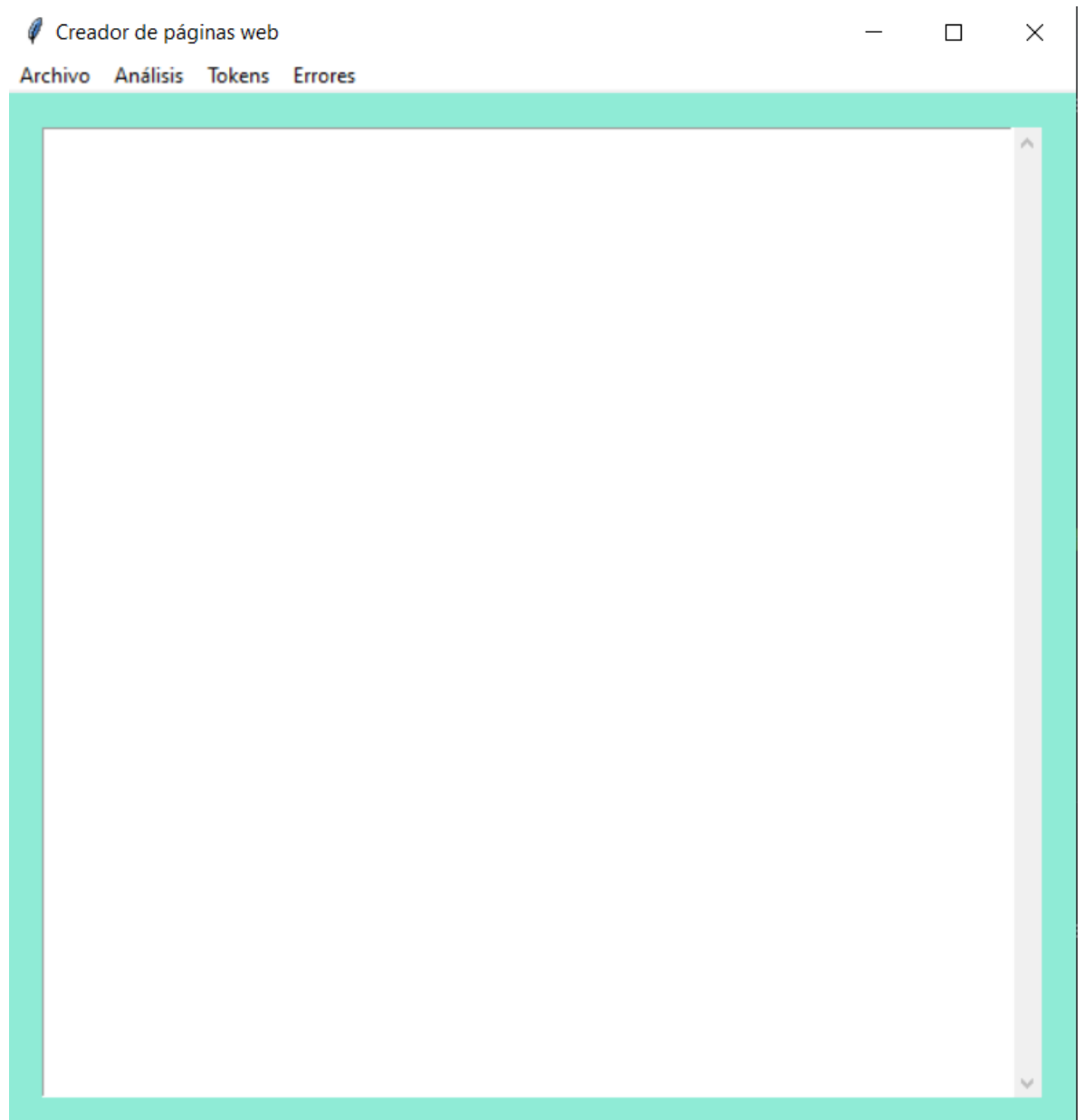
Se utilizó el Paradigma de Programación Orientado a Objetos debido a la capacidad de abstracción que se tiene para modelar objetos de la vida real en el código. Asimismo, se cuenta con diversas clases para modular de una mejor forma el código para que así fuera más visible y entendible al momento de tener la necesidad de realizar un cambio.

Para la lógica del programa se aplicó la teoría del método del árbol para determinar el autómata finito determinista. Así como el uso de gramáticas libres de contexto y autómatas de pila.

Adicionalmente, se utilizó la biblioteca Tkinter debido a que viene por defecto en las bibliotecas de Python y ofrece muchas opciones para crear una buena interfaz para los usuarios.

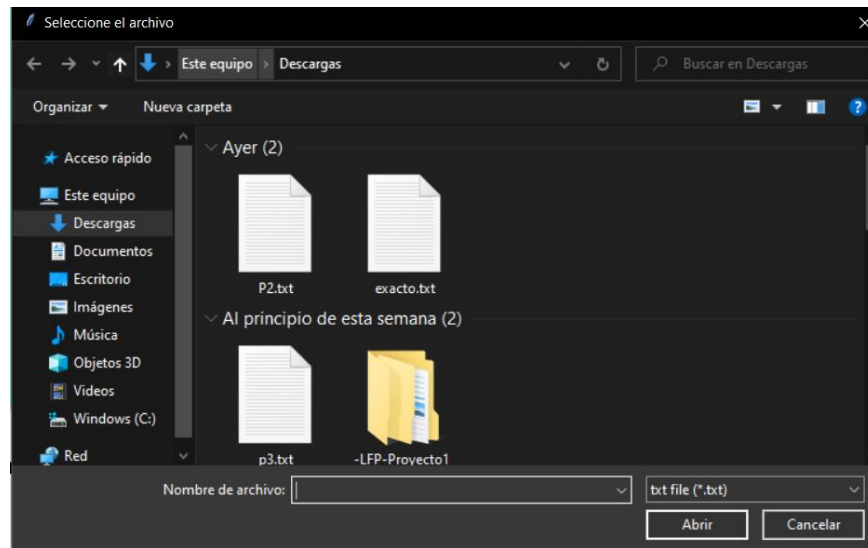
INTERFACES PRINCIPALES

- **Pantalla de inicio**



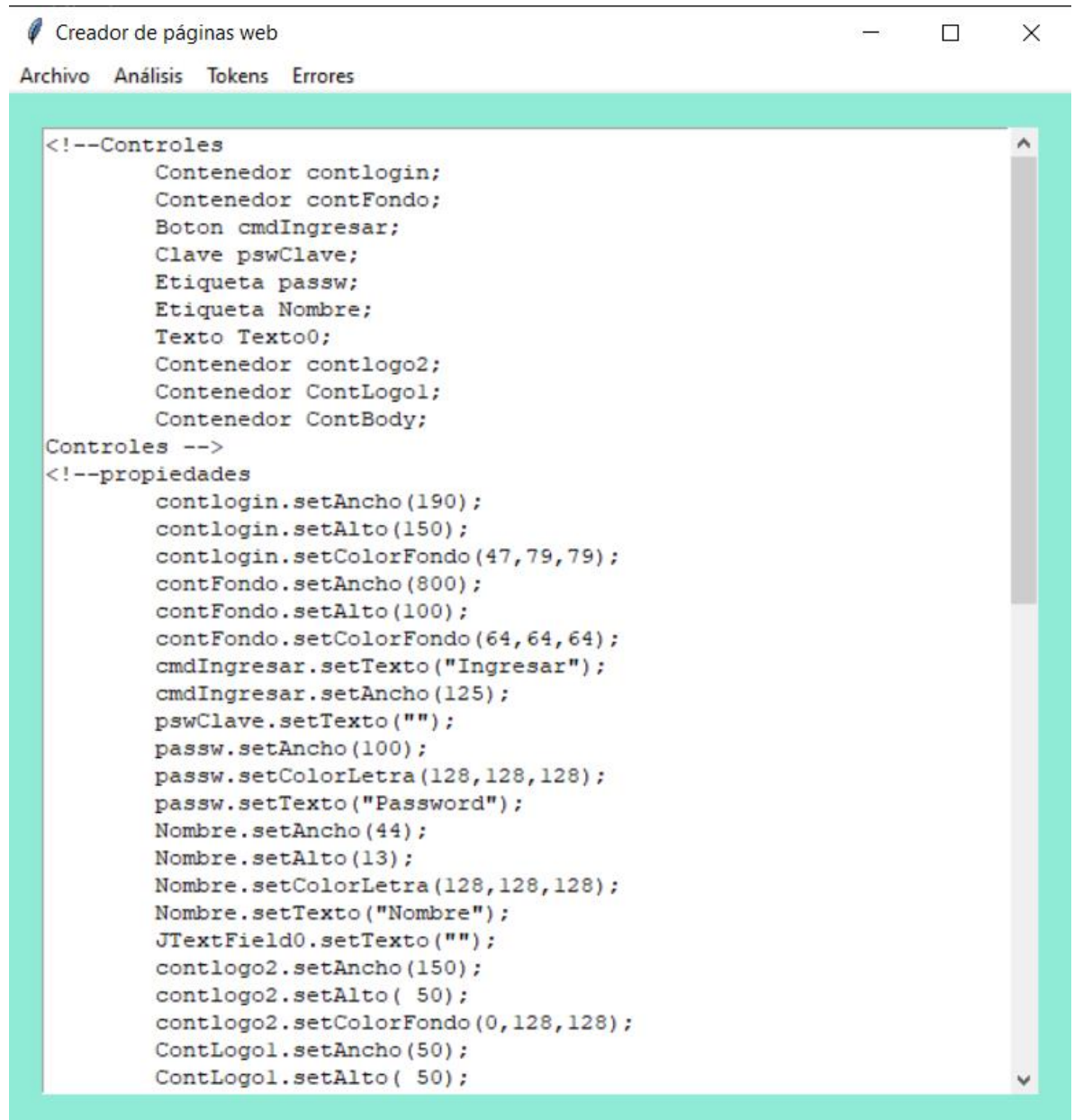
Esta ventana muestra el área de texto en la cual se adjuntará el contenido de un archivo al momento de abrirlo. Así, como el menú de barra con los menús *Archivo*, *Análisis*, *Tokens* y *Errores*.

- **Opción abrir archivo**



Esta opción despliega el administrador de archivos y únicamente los archivos con la extensión permitida.

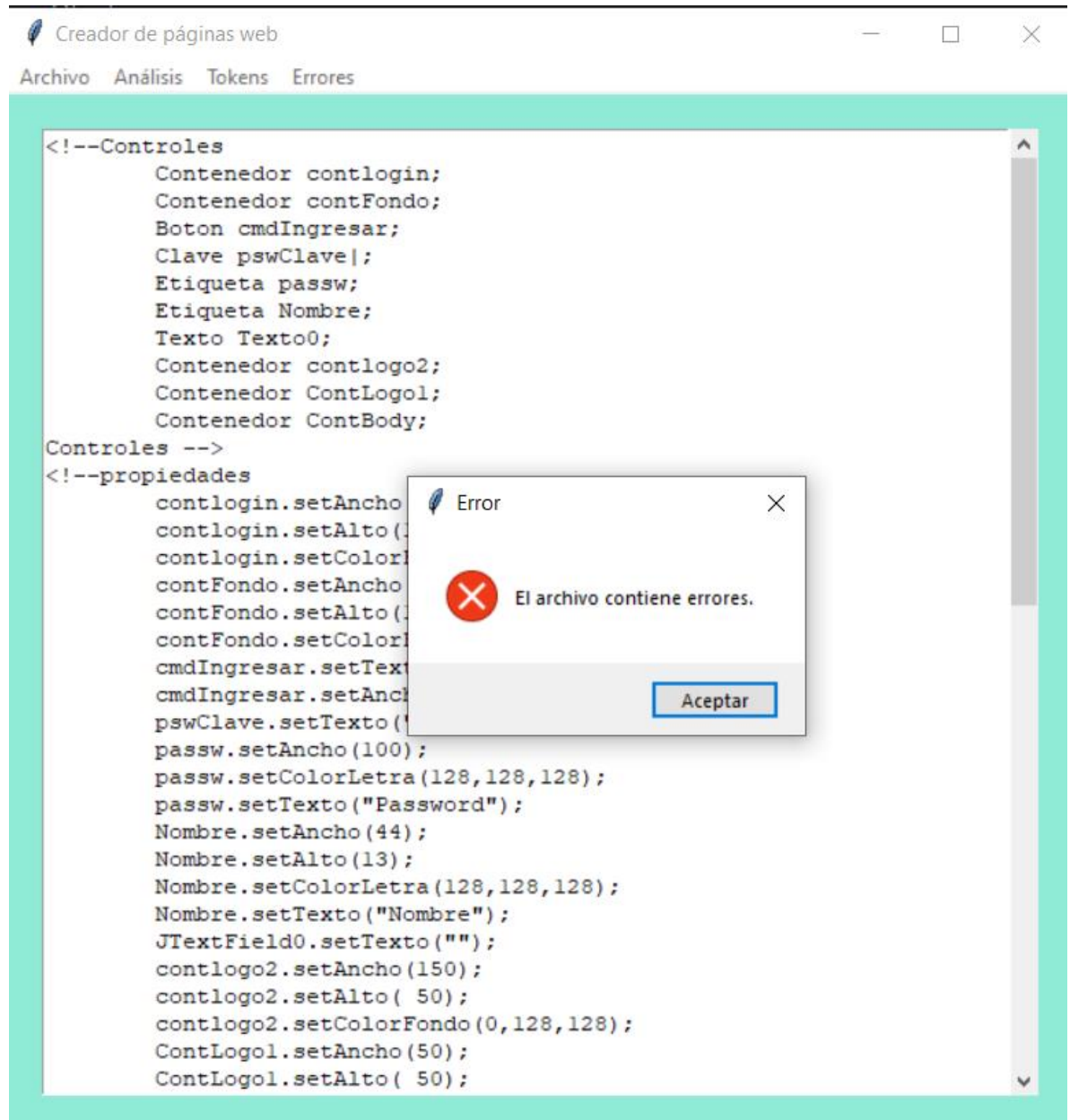
- **Ventana Principal con contenido cargado**



```
<!--Controles
    Contenedor contlogin;
    Contenedor contFondo;
    Boton cmdIngresar;
    Clave pswClave;
    Etiqueta passw;
    Etiqueta Nombre;
    Texto Texto0;
    Contenedor contlogo2;
    Contenedor ContLogol;
    Contenedor ContBody;
Controles -->
<!--propiedades
    contlogin.setAncho(190);
    contlogin.setAlto(150);
    contlogin.setColorFondo(47,79,79);
    contFondo.setAncho(800);
    contFondo.setAlto(100);
    contFondo.setColorFondo(64,64,64);
    cmdIngresar.setTexto("Ingresar");
    cmdIngresar.setAncho(125);
    pswClave.setTexto("");
    passw.setAncho(100);
    passw.setColorLetra(128,128,128);
    passw.setTexto("Password");
    Nombre.setAncho(44);
    Nombre.setAlto(13);
    Nombre.setColorLetra(128,128,128);
    Nombre.setTexto("Nombre");
    JTextField0.setTexto("");
    contlogo2.setAncho(150);
    contlogo2.setAlto( 50);
    contlogo2.setColorFondo(0,128,128);
    ContLogol.setAncho(50);
    ContLogol.setAlto( 50);
```

Al cargar un archivo al sistema este se mostrará de la siguiente forma.

- Opción “Generar Pagina Web” del menú “Análisis”:



Si el archivo ingresado contiene errores de cualquier tipo, se mostrará una alerta indicándolo.

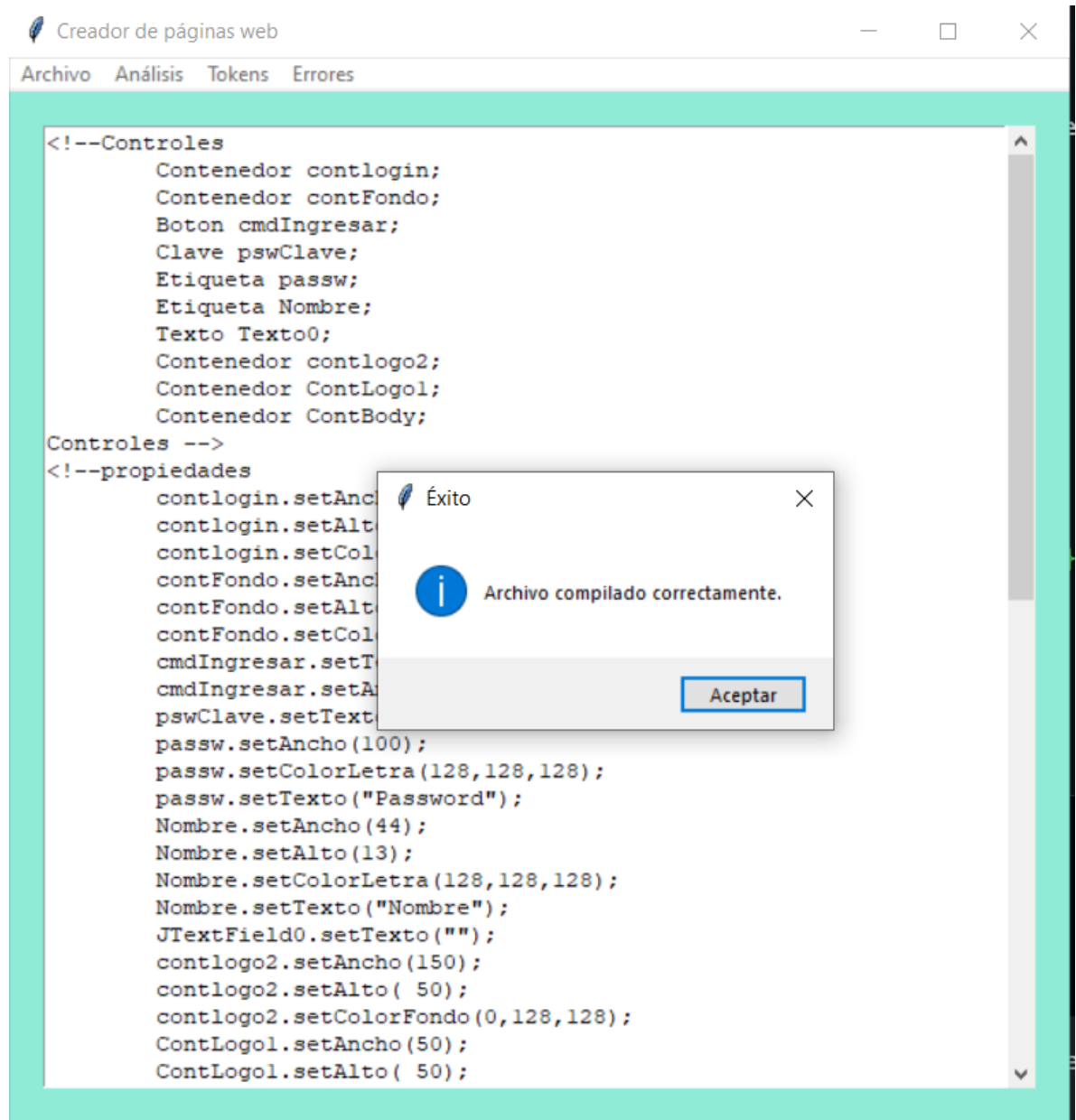
- Opción “Ver Tokens” del menú “Tokens”:

tk			—	□	×
Número	Tipo	Lexema			
12	Mayor que	>			
14	Guion	-			
14	Guion	-			
1	Reservada	Colocacion			
6	Punto y coma	;			
11	Paréntesis derecho)			
7	Valor	145			
9	Coma	,			
7	Valor	680			
10	Paréntesis izquierdo	(

Esta opción despliega una ventana con todos los tokens encontrados dentro del archivo ingresado. Este detalla el número de token, el tipo y el lexema correspondiente a cada uno.

- Opción “Ver Errores” del menú “Errores”:

- Opción “Generar Pagina Web” del menú “Análisis”:



De no tener ningún tipo de error, el programa lo notificará al usuario por medio de una ventana emergente. Esto significa que la página web fue generada correctamente.

- **Página web generada**



Este es un ejemplo del tipo de páginas web que se pueden generar por medio de este programa de software.

PLANIFICACIÓN O ESTIMACIÓN

- **Analizador léxico:** 2 días
- **Analizador sintáctico:** 2 días
- **Generación de lenguaje HTML y CSS:** 2 días

GLOSARIO

1. Extensión de archivo: Las extensiones indican qué aplicación ha creado el archivo o puede abrirlo, y qué icono se debe utilizar para el archivo. Por ejemplo, la extensión docx indica al equipo que Microsoft Word puede abrir el archivo y que debe mostrar un icono de Word al verlo en el Explorador de archivos.
2. Explorador de archivos: El Explorador de archivos o Explorador de Windows, como fue nombrado hasta la edición de Windows 8, es el administrador de archivos oficial del sistema operativo Microsoft Windows.
3. Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
4. Sistema: En terminología informática el software de sistema, denominado también software de base, consiste en un software que sirve para controlar un proceso.
5. Paradigma: El término paradigma es empleado para indicar un patrón, modelo, ejemplo o arquetipo. Por lo general hace referencia a una serie de teorías que son tomadas como modelo a seguir al momento de solucionar cualquier tipo de problemas que puedan surgir en determinadas situaciones.