Manual Técnico

Generador de código HTML

Fecha: 01 de abril de 2024

Contenido

OBJETIVO

El objetivo de este manual técnico es proporcionar a cualquier persona interesada en el proyecto una guía completa sobre la estructura, funcionamiento y desarrollo del traductor de código HTML. Este manual tiene como propósito facilitar la comprensión de cómo se ha implementado la aplicación, permitiendo su mantenimiento, extensión y personalización de acuerdo con las necesidades cambiantes del usuario.

ALCANCE

Generador de código HTML – Este manual abarca aspectos como la descripción, propósito y funciones disponibles del proyecto, características principales, requisitos mínimos del sistema, instalación y ejecución del mismo, incluyendo cualquier configuración adicional que pueda ser necesaria.

DESARROLLO DEL MANUAL TÉCNICO

Descripción General

El Traductor HTML es una aplicación desarrollada en Python con interfaz gráfica utilizando la biblioteca Tkinter. Su objetivo principal es analizar un texto mediante un archivo de entrada, el cual tiene un formato específico y traducirlo a un documento HTML, a su vez, se genera un reporte con los lexemas y tokens obtenidos mediante un analizador léxico o bien si el archivo posee errores o caracteres inválidos también se genera un reporte de ellos.

Requisitos del Sistema

- Python 3.x instalado en el sistema.
- Bibliotecas necesarias instaladas: tkinter.

Instalación

Debido a que es una aplicación que se ejecuta con interfaz gráfica, se debe clonar o descargar el repositorio del proyecto desde Github para luego abrir el proyecto en el IDE de su preferencia (se recomienda Visual Studio Code) y correr la aplicación.

Descripción del funcionamiento

Interfaz Gráfica

La aplicación utiliza Tkinter para proporcionar una interfaz gráfica simple al usuario. La interfaz incluye:

- Un botón para abrir un archivo de texto.
- Un botón para traducir el texto a HTML.
- Un botón para salir de la aplicación.
- Dos áreas de texto: una para ingresar el texto a traducir y otra para mostrar el resultado HTML.

```
# Crear la ventana principal
ventana = tk.Tk()
ventana.title("Traductor HTML")
ventana.configure(bg="#BFEA7C")
ventana.geometry("1000x600")

# Contenedor para los botones
frame_botones = tk.Frame(ventana, bg="#BFEA7C")
frame_botones.pack(pady=20)

# Botones
boton_abrir = tk.Button(frame_botones, text="Abrir archivo", command=abrir_archivo, width=15)
boton_abrir.pack(side=tk.LEFT, padx=10)

boton_traducir = tk.Button(frame_botones, text="Traducir a HTML", command=analizar_texto_y_most
boton_traducir.pack(side=tk.LEFT, padx=10)

boton_salir = tk.Button(frame_botones, text="Salir", command=salir, width=15)
boton_salir.pack(side=tk.LEFT, padx=10)

# Contenedor para los TextAreas
frame_textareas = tk.Frame(ventana, bg="#BFEA7C")
frame_textareas.pack(expand=True, fill=tk.BOTH, padx=20, pady=20)
# Primer textarea
textAreaInicial = tk.Text(frame_textareas, height=20, width=40)
textAreaInicial.pack(side=tk.LEFT, padx=10, pady=10, expand=True, fill=tk.BOTH)

# Segundo textarea
textAreaFinal = tk.Text(frame_textareas, height=20, width=40, state="normal")
textAreaFinal.pack(side=tk.LEFT, padx=10, pady=10, expand=True, fill=tk.BOTH)
```

Análisis Léxico

El análisis de texto se realiza mediante la función analizar_texto, que toma el texto ingresado por el usuario y busca patrones específicos utilizando expresiones regulares.

Se identifican tokens con sus valores correspondientes, como:

- Inicio
    - Encabezado
        - TituloPagina
    - Cuerpo
        - Titulo
            - Texto
            - Posición
            - Tamaño

        - Color
    - Fondo
        - Color
    - Párrafo
        - Texto
        - Posición
    - Texto
        - Fuente
        - Color
        - Tamaño
    - Código
        - Texto
        - Posición
    - Negrita, subrayado, tachado y cursiva
        - Texto
    - Salto
        - cantidad
    - Tabla
        - Filas
        - Columnas
        - Elemento

```
4   def analizadorLexico(textAreaInicial, textAreaFinal):
5       lexemas = []
6       errores = []
7       palabra = ""
8       dentro_cadena = False
9
10      # Obtener el texto del text area
11      texto = textAreaInicial.get("1.0", "end")
12
13      # Inicializar las variables de fila y columna
14      fila = 1
15      columna = 0
16
17      # Iterar sobre cada caracter del texto
18      for char in texto:
19          columna += 1
20          # Se verifica si está dentro de una cadena de texto
21          if char == '"':
22              dentro_cadena = not dentro_cadena
23              palabra += char
24              continue
```

```
        # Si está dentro de una cadena de texto, se añade el caracter a la palabra
        if dentro_cadena:
            palabra += char
            continue

        # Se verifica que el caracter sea una letra, un dígito o un carácter especial
        if char.isalnum():
            palabra += char
        else:
            if palabra:
                # Verificar si la palabra es una palabra reservada
                if palabra in ['Inicio', 'Encabezado', 'TituloPagina', 'Cuerpo', 'Titulo', 'tex
                    lexemas.append(Lexema("Palabra Reservada", palabra, fila, columna - len(pal
                elif palabra.isdigit():
                    lexemas.append(Lexema("Número", palabra, fila, columna - len(palabra)))
                else:
                    lexemas.append(Lexema("Cadena", palabra, fila, columna - len(palabra)))
                palabra = ""
```

```
                # Otros caracteres especiales
                if char in [',']:
                    lexemas.append(Lexema("Coma", char, fila, columna))
                elif char in ['.']:
                    lexemas.append(Lexema("Punto", char, fila, columna))
                elif char in ['{']:
                    lexemas.append(Lexema("Llave de apertura", char, fila, columna))
                elif char in ['}']:
                    lexemas.append(Lexema("Llave de cierre", char, fila, columna))
                elif char in [':']:
                    lexemas.append(Lexema("Dos puntos", char, fila, columna))
                elif char in ['[']:
                    lexemas.append(Lexema("Corchete de apertura", char, fila, columna))
                elif char in [']']:
                    lexemas.append(Lexema("Corchete de cierre", char, fila, columna))
                elif char in ['=']:
                    lexemas.append(Lexema("Igual", char, fila, columna))
                elif char in [';']:
                    lexemas.append(Lexema("Punto y coma", char, fila, columna))
                elif char in [' ']:
                    continue
                elif char == '\n':
                    fila += 1
                    columna = 0  # Reiniciar la columna cuando se encuentra un salto de línea
```

```
    # Verificar si hay una palabra aún por agregar
    if palabra:
        if palabra in ['Inicio', 'Encabezado', 'TituloPagina', 'Cuerpo', 'Titulo', 'texto', 'po
            lexemas.append(Lexema("Palabra Reservada", palabra, fila, columna - len(palabra)))
        elif palabra.isdigit():
            lexemas.append(Lexema("Número", palabra, fila, columna - len(palabra)))
        else:
            lexemas.append(Lexema("Cadena", palabra, fila, columna - len(palabra)))

    imprimirLexemasYErrores(lexemas, errores)
    return lexemas, errores
```

Autómata Finito Determinista

El AFD desempeña un papel fundamental en el análisis léxico del texto de entrada, ya que permite reconocer y clasificar los diferentes tokens que conforman el lenguaje específico utilizado por el traductor.

El AFD implementado en el proyecto del Traductor HTML consta de un conjunto finito de estados, un alfabeto de entrada, una función de transición y un conjunto de estados de aceptación. A continuación, se describen los componentes principales del AFD:

Conjunto de Estados (N)

- El conjunto de estados del AFD representa los diferentes estados en los que puede encontrarse el autómata durante el proceso de análisis léxico.
- Cada estado representa una etapa específica del proceso de reconocimiento de tokens, desde el estado inicial hasta los estados de aceptación.

$$N = \{S0, S1, S2, S3, S4, S5\}$$

Alfabeto de Entrada (Σ)

- El alfabeto de entrada del AFD incluye todos los caracteres válidos que pueden formar parte del texto de entrada.
- Este alfabeto define los símbolos que el AFD puede reconocer y procesar durante el análisis léxico.

$$\Sigma = \{I, E, T, C, F\}$$

$$\mathrm{I} = \{Inicio\} \quad E = \{Encabezado\} \quad T = \{TituloPagina\} \quad C = \{Cuerpo\}$$

$$F = \begin{Bmatrix} Titulo, Fondo, Parrafo, Texto, Codigo, Negrita, Subrayado, \\ Tachado, Cursiva, Salto, Tabla \end{Bmatrix}$$

Estado Inicial (S0):

El estado inicial del AFD representa el punto de partida del autómata en el proceso de análisis léxico.

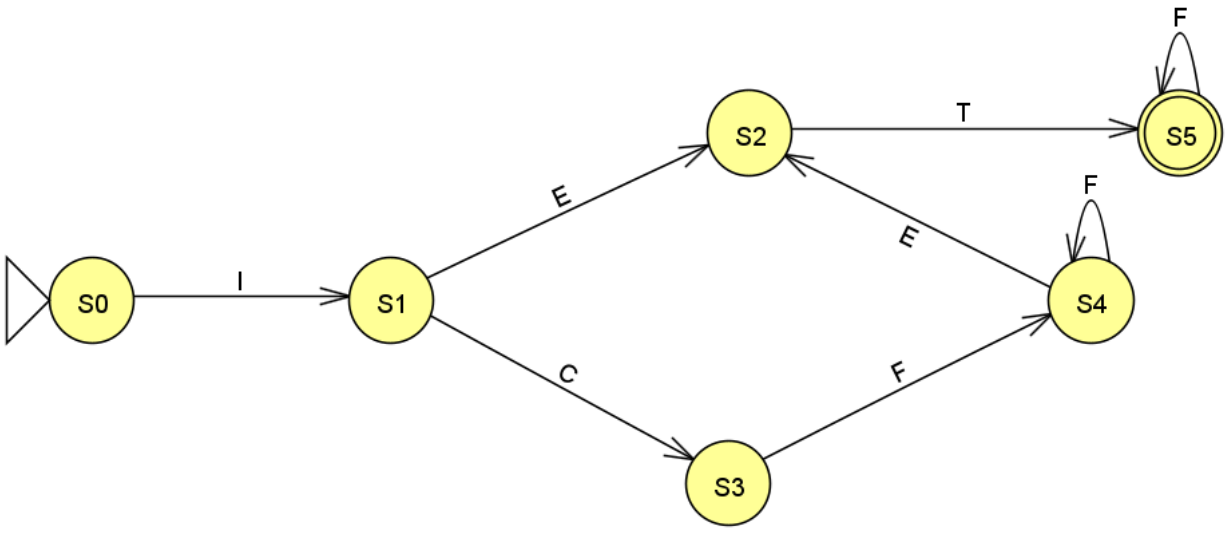
Desde el estado inicial, el AFD comienza a procesar el texto de entrada y a reconocer los tokens presentes en él.

$$Inicio = S0$$

Función de Transición

- La función de transición del AFD define cómo el autómata se mueve de un estado a otro en respuesta a la entrada proporcionada.
- Para cada estado y símbolo de entrada, la función de transición determina el próximo estado al que el AFD debe transicionar.

S0 → IS1
S1 → ES2 | CS3
S2 → TS5
S3 → FS4
S4 → ES2 | FS4
S5 → FS5 | ε



Traducción a HTML

La traducción a HTML se lleva a cabo mediante la función traducir_a_html, que toma los tokens identificados en el análisis léxico y de texto y genera un documento HTML correspondiente.

Los diferentes elementos del texto, como título, párrafos, código, etc., se traducen a etiquetas HTML apropiadas.

Se aplican estilos CSS correspondientes, como color de texto, tamaño de fuente, alineación, etc.

```
22  def traducir_a_html(tokens):
23      html = "<!DOCTYPE html>\n<html>\n<head>\n<title>"
24      titulo_pagina = ""
25      cuerpo = ""
26      fondo_estilo = ""
27      saltos = 0
28
29      # Tamaños de título a etiquetas HTML
30      tamanos_a_encabezados = {
31          't1': 'h1',
32          't2': 'h2',
33          't3': 'h3',
34          't4': 'h4',
35          't5': 'h5',
36          't6': 'h6'
37      }
```

```
39      for token, valor in tokens:
40          print(f"Token: {token}, Valor: {valor}")  # Imprimir el token y su valor
41
42          if token == 'TITULOPAGINA':
43              titulo_pagina = valor
44          elif token == 'FONDO':
45              fondo_estilo = f"background-color:{convertir_color_a_hexadecimal(extraer_valor(valo
46          elif token == 'TITULO':
47              alineacion = extraer_valor(valor, 'posicion', 'left')
48              alineacion = 'left' if alineacion == 'izquierda' else alineacion
49              alineacion = 'center' if alineacion == 'centro' else alineacion
50              alineacion = 'right' if alineacion == 'derecha' else alineacion
51              # Pasar el tamaño del titulo a la etiqueta HTML
52              tamaño = extraer_valor(valor, 'tamaño', 't1')
53              etiqueta_encabezado = tamanos_a_encabezados.get(tamaño, 'h1')
54              cuerpo += f"<{etiqueta_encabezado} style='color:{convertir_color_a_hexadecimal(extr
55          elif token == 'PARRAFO':
56              alineacion = extraer_valor(valor, 'posicion', 'left')
57              alineacion = 'left' if alineacion == 'izquierda' else alineacion
58              alineacion = 'center' if alineacion == 'centro' else alineacion
59              alineacion = 'right' if alineacion == 'derecha' else alineacion
60              cuerpo += f"<p style='color:{convertir_color_a_hexadecimal(extraer_valor(valor, 'co
```

```
        elif token == 'TEXTO':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            alineacion = 'center' if alineacion == 'centro' else alineacion
            alineacion = 'right' if alineacion == 'derecha' else alineacion
            cuerpo += f"<span style='font-family:{extraer_valor(valor, 'fuente')}; color:{conve
        elif token == 'CODIGO':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            alineacion = 'center' if alineacion == 'centro' else alineacion
            alineacion = 'right' if alineacion == 'derecha' else alineacion
            cuerpo += f"<code style='color:{convertir_color_a_hexadecimal(extraer_valor(valor,
        elif token == 'NEGRITA':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            cuerpo += f"<b style='color:{convertir_color_a_hexadecimal(extraer_valor(valor, 'co
        elif token == 'SUBRAYADO':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            cuerpo += f"<u style='color:{convertir_color_a_hexadecimal(extraer_valor(valor, 'co
```

```
        elif token == 'TACHADO':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            cuerpo += f"<strike style='color:{convertir_color_a_hexadecimal(extraer_valor(valor
        elif token == 'CURSIVA':
            alineacion = extraer_valor(valor, 'posicion', 'left')
            alineacion = 'left' if alineacion == 'izquierda' else alineacion
            cuerpo += f"<i style='color:{convertir_color_a_hexadecimal(extraer_valor(valor, 'co
        elif token == 'TABLA':
            filas = int(extraer_valor(valor, 'filas', '0'))
            columnas = int(extraer_valor(valor, 'columnas', '0'))
            tabla_html = "<table border='1'>\n"
            elementos_tabla = extraerElementosTabla(valor)
            for fila in range(filas):
                tabla_html += "<tr>\n"
                for columna in range(columnas):
                    texto_elemento = ""
                    for elemento in elementos_tabla:
                        if int(elemento['fila']) == fila + 1 and int(elemento['columna']) == co
                            texto_elemento = elemento['texto']
                            break
                    tabla_html += f"<td>{texto_elemento}</td>\n"
                tabla_html += "</tr>\n"
```

```
            tabla_html += "</table>\n"
            cuerpo += tabla_html
        elif token == 'SALTO':
            cantidad_saltos = int(extraer_valor(valor, 'cantidad', '1'))
            print(f"SALTOS DE LÍNEA: {cantidad_saltos}")  # verificar la cantidad de saltos de
            cuerpo += "<br>" * cantidad_saltos
            print(f"CUERPO DESPUÉS DEL SALTO: {cuerpo}")  # verificar después de agregar los sa

    html += f"{titulo_pagina}</title>\n</head>\n<body style='{fondo_estilo}'>\n{cuerpo}</body>\
```

Generación de Resultados

El resultado HTML generado se muestra en el área de texto final para que el usuario pueda ver y copiar.

Se generan archivos HTML adicionales para registrar los lexemas y errores encontrados durante el análisis léxico.

```
85  def imprimirLexemasYErrores(lexemas, errores):
86      # Escribir los errores en un archivo HTML
87      with open("errores.html", "w", encoding='utf-8') as f:
88          f.write("<html>\n<head>\n<title>Errores Léxicos</title>\n</head>\n<body>\n")
89          f.write("<h1>Errores Léxicos</h1>\n")
90          f.write("<table border='1'>\n")
91          f.write("<tr><th>Caracter Inválido</th><th>Cantidad</th><th>Fila</th><th>Columna</th></
92          caracteres_invalidos = {}
93          for error in errores:
94              if error.mensaje not in caracteres_invalidos:
95                  caracteres_invalidos[error.mensaje] = {"cantidad": 1, "fila": error.fila, "colu
96              else:
97                  caracteres_invalidos[error.mensaje]["cantidad"] += 1
98          for caracter, info in caracteres_invalidos.items():
99              f.write(f"<tr><td>{caracter}</td><td>{info['cantidad']}</td><td>{info['fila']}</td>
100         f.write("</table>\n")
101         f.write("</body>\n</html>")
102
103         # Escribir los lexemas en un archivo HTML
104     with open("lexemas.html", "w", encoding='utf-8') as f:
105         f.write("<html>\n<head>\n<title>Listado de Tokens y Lexemas</title>\n</head>\n<body>\n"
106         f.write("<h1>Listado de Tokens y Lexemas</h1>\n")
107         f.write("<table border='1'>\n")
108         f.write("<tr><th>Token</th><th>Lexema</th><th>Línea</th><th>Columna</th></tr>\n")
109         for lexema in lexemas:
110             f.write(f"<tr><td>{lexema.tipo}</td><td>{lexema.valor}</td><td>{lexema.linea}</td><
```

Estructura del proyecto

El proyecto sigue una estructura típica de Python:

- Traductor
    - main.py: Script principal para ejecutar la aplicación.
        - lexema.py: contiene las clases Lexema y Error
        - analizador_lexico.py: se analizan los caracteres válidos del lenguaje
        - traductor_html.py: se gestiona la traducción HTML del archivo de entrada.

Manejo de Errores

- Detección de Errores en el Código Fuente:

Durante el desarrollo del proyecto, se emplean buenas prácticas de programación para evitar la introducción de errores en el código.

Se realizan pruebas exhaustivas y revisiones de código para identificar y corregir posibles errores antes de la implementación final del software.

- Manejo de Excepciones:

Se utilizan bloques try-except para manejar excepciones que puedan ocurrir durante la ejecución del programa.

En áreas críticas del código, se implementan bloques de manejo de excepciones para capturar y gestionar posibles errores de forma adecuada.

Áreas de Mejoras Futuras

Se considerará en el futuro mejorar la opción para que al momento de abrir un archivo con caracteres inválidos, no se muestre ninguna traducción si no que solo se cree el reporte de los mismos.