Manual de Tecnico

Para resolver este Proyecto se utilizó un DFA (Autómata Finito Determinista) para poder hacer el análisis léxico del documento y se uso un diagrama de arbol para poder hacer el analizador sintactico.

Para la interfaz grafica se utilizo Tkinter, se utilizaron 7 clases para darle solución al problema la clase Analizador_lexico esta nuestro DFA donde se hace el análisis del documento para poder sacar los tokens respectivos que luego pasaran a la clase Analizador sintactico para poder sacar la información necesaria, la clase Create_html_css se encarga de hacer las paginas HTML y CSS de los datos obtenidos respectivamente de los analizadores, y la nformación necesaria para la elaboración de la pagina web tanto como sus ajustes de estilos etc… ,la clase tokens guarda los tokens no validos que el autómata no reconoce para luego posteriormente si tiene errores el documento los mostrara en la pagina web Se utilizó programación orientada a objetos para poder guardar la información ya analizada del documento.

Clase Analizador_lexico

```python
from Model.tokens import Tokens
from Model.html import HTML
from Model.database import data
class scanner():
    def __init__(self) -> None:
        self.letras = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t",
        self.simbolos_comentarios = [',','#','!','(',')','[',']','{','}','$','%','^',]
        self.numeros = ["1","2","3","4","5","6","7","8","9","0"]
        self.tipo = ["contenedor","boton","clave","etiqueta","texto","areatexto","radioboton","check"]
        self.__data_tokens = data()
        self._data_HTML = data()
        self.__data_tokens_malos_lexicos = data()
        self.__tokens_comentarios = data()

    def analisis_lexico(self,cadena):...

    def Guardar_datos_HTML(self,contenedor,ID):
        token = HTML(contenedor,ID)
        self._data_HTML.insertar_dato(token)
    def Guardar_Tokens(self,dato,fila):
        token = Tokens(dato,fila)
        self.__data_tokens.insertar_dato(token)
    def Guardar_Tokens_malos(self,dato,fila):
        token = Tokens(dato,fila)
        self.__data_tokens_malos_lexicos.insertar_dato(token)
    def Guardar_Tokens_comentarios(self,dato):
        token = Tokens(dato,dato)
        self.__tokens_comentarios.insertar_dato(token)

    def Get_data_tokens(self):
        return self.__data_tokens
    def Get_data_tokens_comentarios(self):
        return self.__tokens_comentarios
    def Get_data_tokens_malos(self):
        resul = self.__data_tokens_malos_lexicos
        return resul
    def Get_data_HTML(self):
        return self._data_HTML
```

Clase Analizador_sintactico

```python
from tokenize import String
from Model.database import data
from Model.CSS import css
from Model.tokens import Tokens
class parser():
    def __init__(self) -> None:
        self.list_ID_propiedad = ["setColorLetra","setTexto","setAlineacion","setColorFondo","setMarcad
        self.token = ""
        self.data_css = data()
        self.data_tokens_malos = data()
        self.ID = ""
        self.propiedad = ""
        self.valor = ""
        self.RGB = {}
        self.posicion = {}
        self.asd = 0
        self.char = ""
        self.fila = 1
        pass
    def analisis_sintactico(self,_list_html,_list_tokens):...

    def Guardar_dato_css(self,ID,propiedad,valor):
        CSs = css(ID,propiedad,valor)
        self.data_css.insertar_dato(CSs)
    def Guardar_dato_tokens_malos(self,tokens,fila):
        token = Tokens(tokens,fila)
        self.data_tokens_malos.insertar_dato(token)
    def Get_tokens_sintactico(self):
        return self.data_css
    def Get_tokens_malos_sintactico(self):
        return self.data_tokens_malos
```

Clase Create_HTML_CSS

```python
class create():
    def __init__(self,_list_html,_list_css) -> None:
        self.list_css = _list_css
        self.list_html = _list_html
        pass
    def create_html(self):
        resul_HTML_final = """"""
        resul_HTML_final2 = """"""
        def obtener_datos_Eti(iter3):...
        def obtener_datos_Boton(iter3):...
        def obtener_datos_check(iter3):...
        def obtener_datos_radio(iter3):...
        def obtener_datos_Texto(iter3):...
        def obtener_datos_AreaTexto(iter3):...
        def obtener_datos_Clave(iter3):...
        def comprobacion(comproba):...
        def html_add(iter):...
        for iter in self.list_css.data:...
        code_html = """..."""
        asd = open("./documentation/index.html","w")
        asd.write(code_html)

    def create_css(self):...
```