

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
Ing. Mario Bautista
Aux. Luis Mazariegos
Aux. Jose Puac



Edgar Daniel Cil Peñate
201503600

Manual Técnico

Descripción del Problema

En la Escuela de Ciencias y Sistemas se propuso un proyecto de un intérprete que englobe los principales lenguajes orientados al diseño web: HTML, CSS y Javascript. Por lo cual, se desarrolló un prototipo para la detección de errores léxicos dentro de la estructura de un proyecto web, dado que al momento de programar en los lenguajes mencionados anteriormente existen caracteres que no son reconocidos por el lenguaje y estos deben ser reportados.

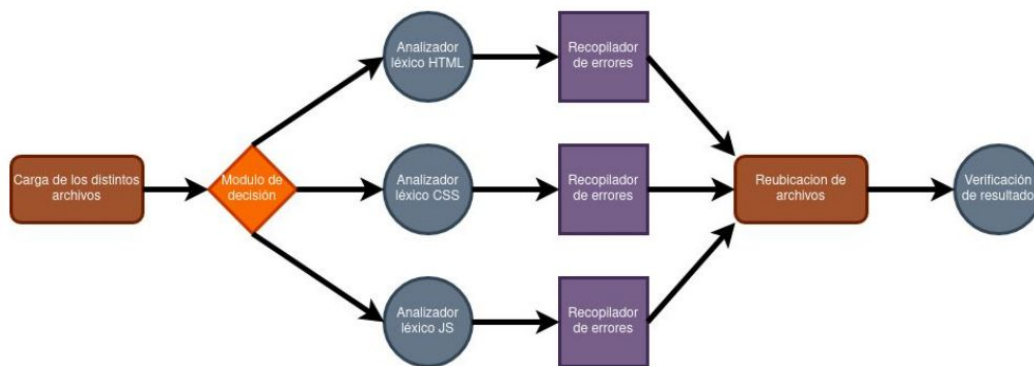
La idea principal del proyecto es que se genere un análisis léxico de cada uno de los distintos archivos de entrada los cuales pueden ser de tipo Marcado como en el caso de HTML, de tipo diseño como CSS o de tipo interpretado como JavaScript.

Se realizó un analizador léxico por lenguaje ya que en cada uno de ellos existen distintos tipos de caracteres reconocidos. Y como salida se espera la construcción del

proyecto en directorios los cuales estarán identificados por medio de paths las cuales serán tomadas y almacenadas por cada archivo dentro de la ejecución del programa, reportando todos los errores léxicos encontrados.

Cabe mencionar que al momento de reconocer un error léxico el análisis no debe detenerse por ningún motivo, sino que debe tener la capacidad de recuperarse y seguir analizando. La idea es lograr obtener todos los errores.

Flujo de la aplicación



Analizador léxico para JavaScript

```
class AnalizadorLexicoJson:
    def __init__(self, entrada, conso, name):
        self.entrada = entrada
        self.estado = 0
        self.linea = 0
        self.columna = 0
        self.consola = conso
        self.nombre = name
        self.ids = {}
        # END

    def analizarJson(self):
        self.linea = 1
        self.columna = 0
        self.estado = 0
        self.consola.clear()
        Errores.error.clear()
        if (len(self.entrada) > 0):
            self.insertText(
                "*****\n\tCOMENZANDO EL ANALISIS\n*****"
            )
            self.inicio()
            self.insertText(
                "*****\n\tFINALIZO EL ANALISIS\n*****"
            )
        else:
            self.insertText("Error: No hay texto por analizar\n")
```

Ln 467, Col 62 Spaces: 4 UTF-8 LF Python Go Live

Para el analizador léxico de JavaScript se implementó la clase ***AnalizadorLexicoJson*** en la cual se definen los métodos que a continuación se describen.

- **Método init()**

Este método es el constructor de la clase y recibe como parámetros los siguientes:

- Entrada: String del texto que se analizará
- Conso: Variable de referencia hacia el componente *consola* para imprimir los errores encontrados.
- Name: Nombre del archivo que se generará posteriormente.

- **Método analizarJson()**

Este método se implementa para comenzar el análisis léxico del texto ingresado y antes de comenzar con el análisis se inicializan las variables utilizadas (línea, columna, error, etc.)

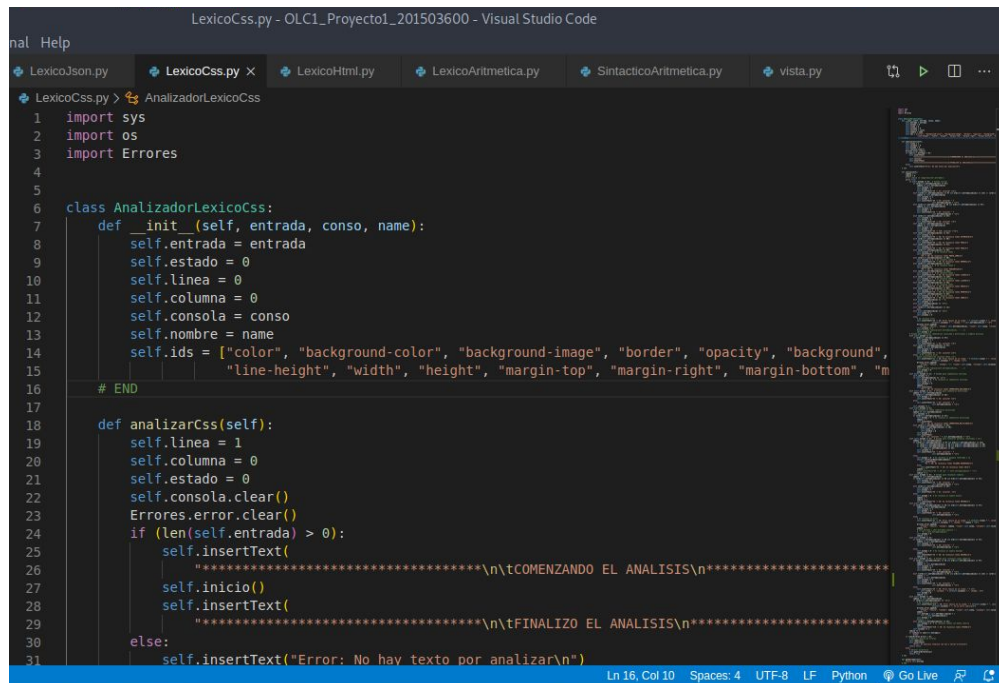
- **Método inicio()**

Este método es el propio analizador en el cual se lee caracter por caracter el texto a analizar, a la vez que se implementan estados para reconocer los distintos tipos de token a reconocer.

- **Método repError()**

Este método genera un archivo html en la ruta **/Desktop/Reportes/** el cual contendrá la información de los errores léxicos encontrados en el archivo de javascript.

Analizador léxico para CSS



```
LexicoCss.py - OLC1_Proyecto1_201503600 - Visual Studio Code
LexicoJson.py LexicoCss.py X LexicoHtml.py LexicoAritmetica.py SintacticoAritmetica.py vista.py
LexicoCss.py > AnalizadorLexicoCss
1 import sys
2 import os
3 import Errores
4
5
6 class AnalizadorLexicoCss:
7     def __init__(self, entrada, conso, name):
8         self.entrada = entrada
9         self.estado = 0
10        self.linea = 0
11        self.columna = 0
12        self.conso = conso
13        self.nombre = name
14        self.ids = ["color", "background-color", "background-image", "border", "opacity", "background",
15                   "line-height", "width", "height", "margin-top", "margin-right", "margin-bottom", "m
16        # END
17
18    def analizarCss(self):
19        self.linea = 1
20        self.columna = 0
21        self.estado = 0
22        self.conso.clear()
23        Errores.error.clear()
24        if (len(self.entrada) > 0):
25            self.insertText(
26                "*****\n\tCOMENZANDO EL ANALISIS\n*****"
27            )
28            self.inicio()
29            self.insertText(
30                "*****\n\tFINALIZO EL ANALISIS\n*****"
31            )
32        else:
33            self.insertText("Error: No hay texto por analizar\n")
```

Para el analizador léxico de CSS se implementó la clase ***AnalizadorLexicoCss*** en la cual se definen los métodos que a continuación se describen.

- **Método init()**

Este método es el constructor de la clase y recibe como parámetros los siguientes:

- Entrada: String del texto que se analizará
- Conso: Variable de referencia hacia el componente *consola* para imprimir los errores encontrados y también para mostrar las transiciones entre los estados por los que pasa para reconocer los tokens
- Name: Nombre del archivo que se generará posteriormente.

- **Método analizarCss()**

Este método se implementa para comenzar el análisis léxico del texto ingresado y antes de comenzar con el análisis se inicializan las variables utilizadas (línea, columna, error, etc.)

- **Método inicio()**

Este método es el propio analizador en el cual se lee caracter por caracter el texto a analizar, a la vez que se implementan estados para reconocer los distintos tipos de token a reconocer.

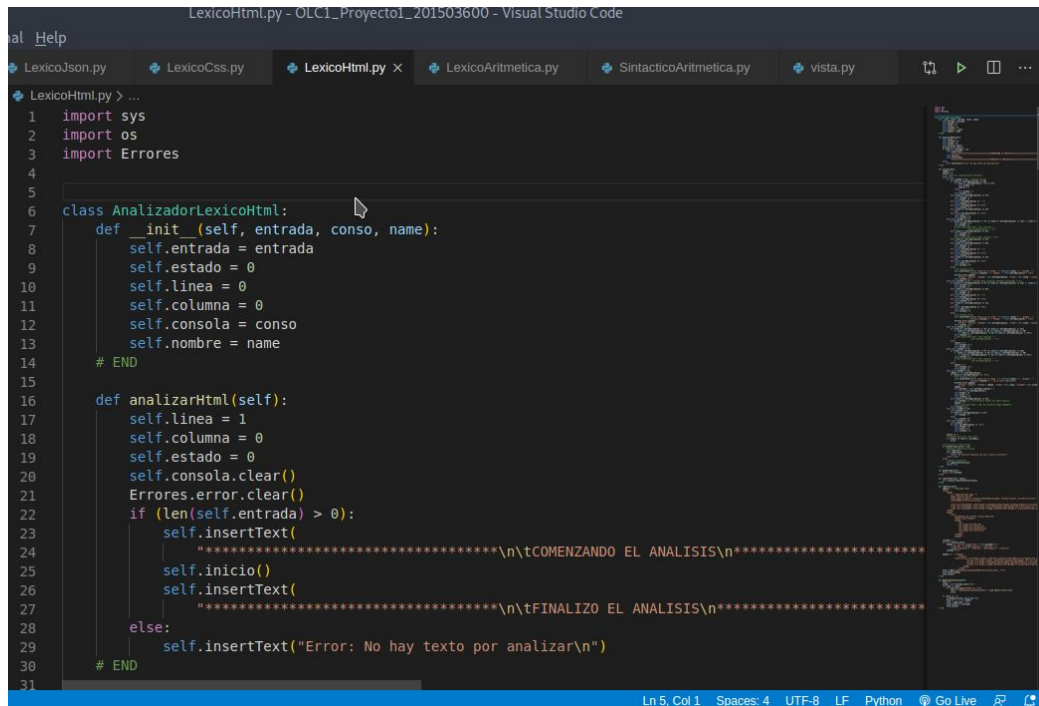
- **Método generarDirectorio()**

En este método se genera el archivo sin errores en la ruta especificada por PATHL.

- **Método repError()**

Este método genera un archivo html en la ruta **/Desktop/Reportes/** el cual contendrá la información de los errores léxicos encontrados en el archivo de css.

Analizador léxico para Html



```
LexicoHtml.py - OLC1_Proyecto1_201503600 - Visual Studio Code
LexicoJson.py LexicoCss.py LexicoHtml.py x LexicoAritmetica.py SintacticoAritmetica.py vista.py
LexicoHtml.py > ...
1 import sys
2 import os
3 import Errores
4
5
6 class AnalizadorLexicoHtml:
7     def __init__(self, entrada, conso, name):
8         self.entrada = entrada
9         self.estado = 0
10        self.linea = 0
11        self.columna = 0
12        self.conso = conso
13        self.nombre = name
14    # END
15
16    def analizarHtml(self):
17        self.linea = 1
18        self.columna = 0
19        self.estado = 0
20        self.conso.clear()
21        Errores.error.clear()
22        if (len(self.entrada) > 0):
23            self.insertText(
24                "*****\n\tCOMENZANDO EL ANALISIS\n*****"
25            )
26            self.inicio()
27            self.insertText(
28                "*****\n\tFINALIZO EL ANALISIS\n*****"
29            )
30        else:
31            self.insertText("Error: No hay texto por analizar\n")
32    # END
```

Para el analizador léxico de Html se implementó la clase ***AnalizadorLexicoHtml*** en la cual se definen los métodos que a continuación se describen.

- **Método init()**

Este método es el constructor de la clase y recibe como parámetros los siguientes:

- Entrada: String del texto que se analizará
- Conso: Variable de referencia hacia el componente *consola* para imprimir los errores encontrados y también para mostrar las transiciones entre los estados por los que pasa para reconocer los tokens
- Name: Nombre del archivo que se generará posteriormente.

- **Método analizarHtml()**

Este método se implementa para comenzar el análisis léxico del texto ingresado y antes de comenzar con el análisis se inicializan las variables utilizadas (línea, columna, error, etc.)

- **Método inicio()**

Este método es el propio analizador en el cual se lee caracter por caracter el texto a analizar, a la vez que se implementan estados para reconocer los distintos tipos de token a reconocer.

- **Método generarDirectorio()**

En este método se genera el archivo sin errores en la ruta especificada por PATHL.

- **Método repError()**

Este método genera un archivo html en la ruta **/Desktop/Reportes/** el cual contendrá la información de los errores léxicos encontrados en el archivo de css.