UNIVESIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

MANUAL TECNICO

OSCAR AUGUSTO PEREZ TZUNUN

201213498

Definición de Interfaz.

```python
import tkinter as tk
import os
from tkinter import filedialog, ttk
from jsscanner import JsScanner
from CssScanner import CssScanner
from HtmlScanner import HtmlScanner
from Reporte import Reporte
from AritmeticScanner import AritmeticScanner
from AritmeticParser import AritmeticParser
from Arbol import Arbol
from Grafo import Grafo


import platform

#Before run execute sudo chmod o+rwx

class Gui():
    def __init__(self, parent):
        self.root = parent
        self.log = None
        self.editor = None
        self.type_file = None
        self.scanner = None
        self.parser = None
        self.expresiones = []
        self.current_file_name = ''
        self.currentPath = ""
        self.create_widgets()

        #necesarias para invocar los distintos analizadores
```

Definición de menu

```
def create_widgets(self):
    self.create_menubar()
    self.create_log()

def create_log(self):
    self.editor = tk.Text(self.root, height = 25)
    self.editor.pack(fill = tk.X, pady = 5)


    self.log = tk.Text(self.root, height = 15, bg = "black", fg = "white")
    self.log.pack(fill = tk.X, pady = 5)

def create_menubar(self):
    menubar = tk.Menu(self.root, tearoff=0)
    filemenu = tk.Menu(menubar, tearoff=0)
    tools = tk.Menu(menubar, tearoff=0)
    report = tk.Menu(menubar, tearoff=0)

    filemenu.add_command(label="Nuevo", command=self.nuevo())
    filemenu.add_command(label="Abrir", command=self.onOpen)
    filemenu.add_command(label = "Abrir Archivo rmt", command=self.onOpenRmt) #escribir la funci
    filemenu.add_command(label="Guardar", command=self.save)
    filemenu.add_command(label="Guardar Como", command=self.saveAs)
    filemenu.add_command(label="Salir", command=self.root.quit)

    tools.add_command(label="Analizar Archivo", command=self.run)
    tools.add_command(label = "Aanlizar Rmt", command=self.runRmt) #add command

    report.add_command(label='Reporte de Tablas', command=self.reporte_tablas)
    report.add_command(label='Reporte Rmt', command=self.reporte_expresiones)
    report.add_command(label='Reporte Arbol', command=self.generar_reporte_arbol)
    report.add_command(label='Reporte Automata', command=self.generar_reporte_grafo)

    menubar.add_cascade(label="Archivo", menu=filemenu)
    menubar.add_cascade(label="Herramientas", menu=tools)
    menubar.add_cascade(label="Reportes", menu=report)
```

Botones

```
def onOpen(self):
    ftypes = [('Css Files', '*.css'), ('JavaScript Files', '*.js'), ('Html Files', '*.html')]
    dialog = filedialog.askopenfilename(initialdir = '~/Escritorio', title='Select File', filetypes = ftypes)


    if not dialog:
        return


    self.type_file = os.path.splitext(dialog)[1]
    self.currentPath = dialog
    self.current_file_name = os.path.basename(dialog)
    self.readFile(dialog)

def onOpenRmt(self):
    type = [('Rmt Files', '*.rmt')]
    dialog = filedialog.askopenfilename(initialdir = '~/Escritorio', title = 'Select File', filetypes = type)

    if not dialog:
        return

    self.readFile(dialog)


def save(self):
    print('save function')
    if self.currentPath != "":
        self.writeFile(self.currentPath, self.editor.get("1.0", "end-1c"))
    else:
        self.saveAs()

def saveAs(self):
    print('save As function')

    ftypes = [('Css Files', '*.css'), ('JavaScript Files', '*.js'), ('Html Files', '*.html')]
    filename = filedialog.asksaveasfile(mode = "w", defaultextension = ".*")

    if filename is None:
```

Escanner HTML

```python
from Token import Token
from Error import Error


class HtmlScanner:

    def __init__(self, texto):
        self.tokens = []
        self.errores = []
        self.text = texto + ' '
        self.out_text = ''
        self.index = -1

    def scanner(self):
        lexema = ''
        linea = 1
        columna = 1
        estado = 0
        index = 0
        fake_index = 0
        tag = False
        found_error = False
        aux_text = self.text
        self.text = self.text.lower()


        while index < len(self.text):
            if estado == 0:
                lexema = ''
                found_error = False
                if self.text[index].isalpha():
                    estado = 1
                    columna += 1
                    lexema += self.text[index]
                elif ord(self.text[index]) == 34:
                    estado = 2
                    columna += 1
                    lexema += self.text[index]
                elif self.text[index].isdigit():
                    estado = 3
```

Escanner CSS

```
from Token import Token
from Error import Error
from io import StringIO


class CssScanner:

    def __init__(self, texto, bitacora):
        self.tokens = []
        self.errores = []
        self.text = texto + ' '
        self.bitacora = bitacora
        self.out_text = StringIO()
        self.index = -1

    def scanner(self):
        lexema = ''
        estado = 0
        linea = 1
        columna = 1
        found_error = False
        aux_text = self.text
        self.text = self.text.lower()

        index = 0
        index_auxiliar = 0
        while index < len(self.text):
            if estado == 0:
                lexema = ''
                found_error = False
                self.bitacora.insert('end-1c', 'Estado 0\n')
                if self.text[index].isalpha():
                    estado = 1
                    columna += 1
                    lexema += self.text[index]
                    self.bitacora.insert('end-1c', 'Reconociendo ID / Reservada ===> Estado 1
                elif self.text[index].isdigit():
                    estado = 2
                    columna += 1
                    lexema += self.text[index]
```

Escanner JavaScript

```python
from Token import Token
from Error import Error
from io import StringIO


class JsScanner:
    def __init__(self, data, t):
        self.tktext = t
        self.text = data + ' '
        self.tokens = []
        self.errores = []
        self.out_text = StringIO()
        self.index = -1
        self.er = {}
        self.out_er = {}

        self.expresiones_regulares()

    def expresiones_regulares(self):
        self.er['id'] = '.L*||LN_'
        self.er['entero'] = '+N'
        self.er['decimal'] = '.+N?.p+N' # p representa el . como lexema no como simbolo
        self.er['cadena'] = '.."*C"'
        self.er['cadena_s'] = "..'*C'"
        self.er['comentario_m'] = '..../@*C@/' # @ representa el * como lexema no como simb
        self.er['comentario_s'] = '..//*C'

    def add_er(self, er):
        if not er in self.out_er:
            self.out_er[er] = self.er[er]


    def scanner(self):
        estado = 0
        lexema = ""
        linea = 1
        columna = 1

        length = len(self.text)
        index = 0
```