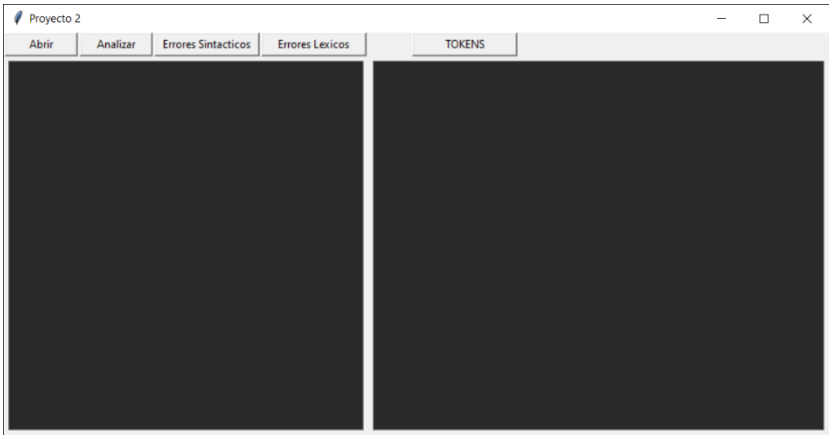
Universidad de san Carlos de Guatemala
Escuela de ingeniería en Ciencias y Sistemas
Facultad de ingeniería

# MANUAL TÉCNICO

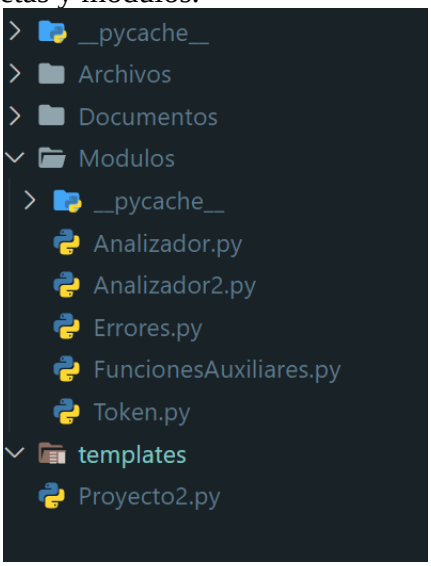
El presente programa es un generador de reportes para la toma de decisiones y facturas que puede ser aplicado a cualquier tipo de negocio, ya que no presenta ningún problema siempre y cuando los datos estén correctamente ingresados. El programa acepta errores léxicos y sintácticos, pero los detecta trata de continuar con su ejecución normal siempre que se pueda. Para ingresar datos previamente cargados desde un archivo, estos tienen que ser con extensión lfp o (.lfp) para que pueda ser reconocido por el programa.

El programa fue corrido en windos 10 y escrito en Visual Estudio Code en el lenguaje Python versión 3.9.2

La apariencia de programa es:




El programa cuenta con las siguientes carpetas y módulos.



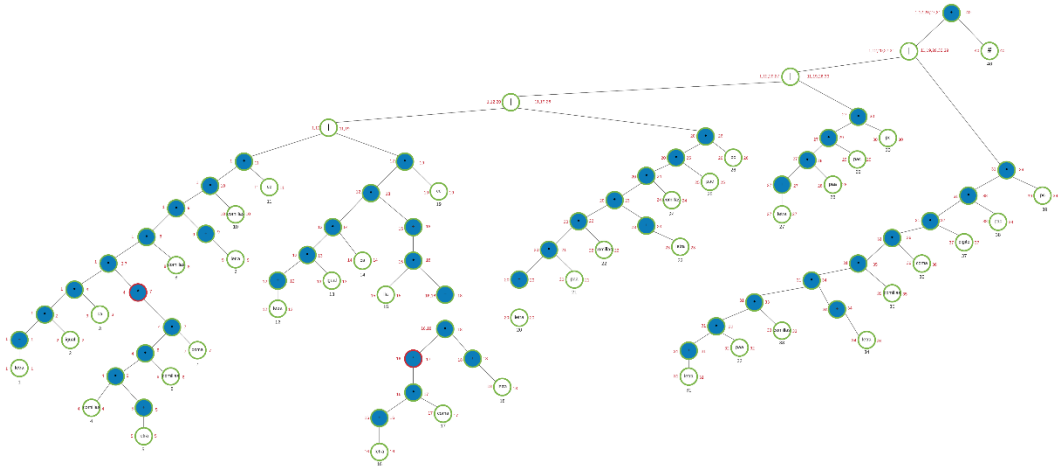

Las principales listas que se usaron para el almacenamiento de datos son:

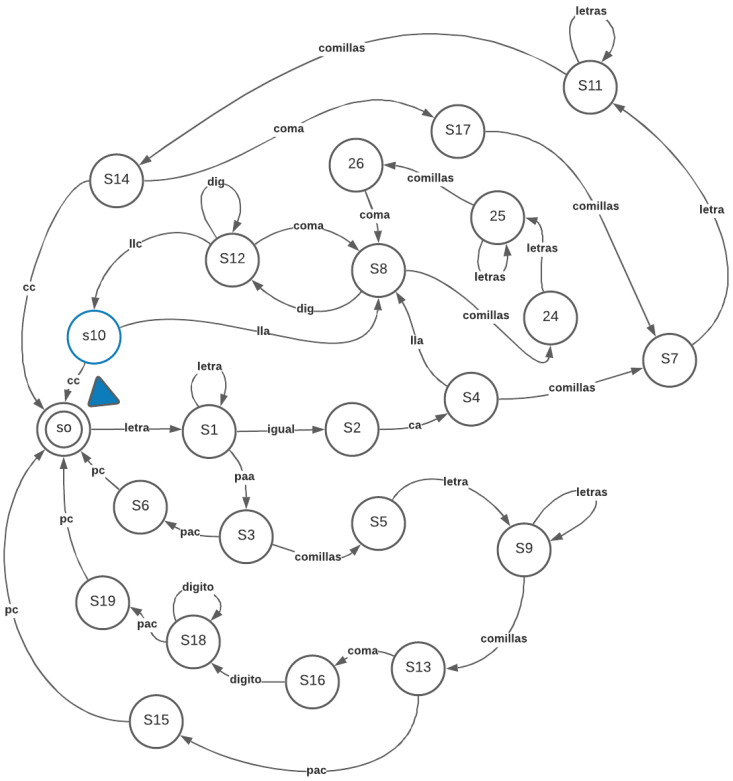
```
self.listaErroresLexicos = []
self.listaErroresSintacticos = []
self.listaTokens = []
self.claves = []
self.registros = []
self.registro = []
```

# LÓGICA DEL PROGRAMA

letra = (A-Z)(a-z)
igual = '='
comillas = "
dig = (0-9)
pc = ";"
ca = [
cc = ]
lla = {
llc = }
paa = (
pac = )
coma = ","
almoadilla = #
comilla = \'

(letra+)(igual)(ca)(  (comillas)(letra+)(comillas)(coma)  )* (comillas)(letra+)(comillas)(cc)
(|)
(letra+)(igual)(ca)(  (lla)(  (letra+)(coma)  )*(letra+))+  (llc))+  (cc)
(|)
(letra+)(paa)(comillas)(letra+)(comillas)(pac)(pc)
(|)
(letra+)(paa)(pac)(pc)
(|)
(letra+)(paa)(comillas)(letra+)(comillas)(coma)(digito)(pac)(pc)





Inicio → Instrucciones
Instrucciones → ListaDeInstrucciones Instrucción
Instrucción → ListaDeIntroducciones Instrucción | ε
ListaDeInstrucciones → InstrClaves | InstrRegistros | InstrImprimir | InstrImprimirLN | InstrConteo | InstrPromedio | InstrContarSi | InstrDatos | InstrSumar | InstrMax | InstrMin | InstrExportar
InstrClaves → TkClaves  Corchete Abierto Comillas clave comillas masDatos corcheteCerrado
masDatos| →  clave
InstrRegistros → Tk Registros Corchete Abierto valor masDatos corchete Cerrado
masDatos2 → comillas valor comillas
InstrConteo → TkConteo CorcheteAbierto CorcheteCerrado Comillas
InstrPromedio → TKPromedio CorcheteAbierto Comillas datoACalcular Comillas CorcheteCerrado comillas
InstrContarSi → TkContarSi CorcheteAbierto Comillas datoACalcular Comillas Coma Dato CorcheteCerrado comillas
InstrDatos → TkDatos CorcheteAbierto CorcheteCerrado Comillas
InstrSumar → TkSumar corcheteAbierto Comillas datoACalcular Comillas Corchete Cerrado
InstrMax → TkMax corcheteAbierto Comillas datoACalcular Comillas Corchete Cerrado
InstrMin → TkMin CorcheteAbierto Comillas datoACalcular Comillas CorcheteCerrado
InstrExportar → TkReporte CorcheteAbierto Comillas nombreReporte Comillas CorcheteCerrado

# CÓDIGO:

```
def analizar(self, cadena):
    self.__init__()
    self.estado = 0
    self.indexCadena = 0
    while self.indexCadena < len(cadena):
        if self.estado == 0:
            self.estado0(cadena[self.indexCadena])
        elif self.estado == 1:
            self.estado1(cadena[self.indexCadena])
        elif self.estado == 2:
            self.estado2(cadena[self.indexCadena])
        elif self.estado == 3:
            self.estado3(cadena[self.indexCadena])
        elif self.estado == 4:
            self.estado4(cadena[self.indexCadena])
        elif self.estado == 5:
            self.estado5(cadena[self.indexCadena])
        elif self.estado == 6:
            self.estado6(cadena[self.indexCadena])
        elif self.estado == 7:
            self.estado7(cadena[self.indexCadena])
        elif self.estado == 8:
            self.estado8(cadena[self.indexCadena])
        elif self.estado == 9:
            self.estado9(cadena[self.indexCadena])
        elif self.estado == 10:
            self.estado10(cadena[self.indexCadena])
        elif self.estado == 11:
            self.estado11(cadena[self.indexCadena])
        elif self.estado == 12:
            self.estado12(cadena[self.indexCadena])
        elif self.estado == 13:
            self.estado13(cadena[self.indexCadena])
        elif self.estado == 14:
            self.estado14(cadena[self.indexCadena])
        elif self.estado == 15:
```

```
def generarTablaTokens(self):
    head = EncabezadoTokens()
    bottom = finalhtml()
    body = ""
    for datos in self.listaTokens:
        body += f"""
                <tr>
                    <td>{datos.enviarData()[0]}</td>
                    <td>{datos.enviarData()[1]}</td>
                    <td>{datos.enviarData()[2]}</td>
                    <td>{datos.enviarData()[3]}</td>
                </tr>"""
    html = open("templates/ListaTokens.html",'w+')
    html.write(head+body+bottom)
```

```
class Errores:

    def __init__(self, descripcion, linea, columna):
        self.descripcion = descripcion
        self.linea = linea
        self.columna = columna

    def imprimirData(self):
        print(self.descripcion,self.linea, self.columna)

    def enviarData(self):
        return [self.descripcion,self.linea, self.columna]
```

```python
def rutaArchivo() -> str:
    """Devuelve la ruta del archivo"""
    archivo = filedialog.askopenfilename(filetypes=(("Archivos lfp","*.lfp"),
    return archivo

def leerArchivo(ruta):
    archivo = open(ruta, 'r')
    contenido = archivo.read()
    archivo.close()
    return contenido

def escribirArchivo(ruta, contenido):
    archivo = open(ruta, 'w')
    archivo.write(contenido)
    archivo.close()
```