UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
PRIMER SEMESTRE 2021

ROBERTO CARLOS GOMEZ DONIS

CARNET: 202000544

MANUAL USUARIO

Guatemala, 20 de marzo del 20222

Tabla de contenido

I. Introducción

Objetivo

En este manual se describe cada clase a detalle, así mismo el contenido de ella. Una breve explicación de cada método creado y utilizado, tanto como la lógica del proyecto.

Requerimientos mínimos:

✓ REQUISITOS DE HARDWARE

- Procesador de 1,6 GHz o superior
- 1 GB de RAM (1,5 GB si se ejecuta en una máquina virtual)
- 10 GB de espacio disponible en el disco duro
- Unidad de disco duro de 5400 rpm
- Tarjeta de vídeo compatible con DirectX 9 con una resolución de pantalla de 1024 x 768 o superior

✓ REQUISITOS DEL SISTEMA

- En Windows 8.1 y Windows Server 2012 R2, se necesita la actualización 2919355 (también disponible a través de Windows Update) para que Visual Studio 2015 se instale correctamente.

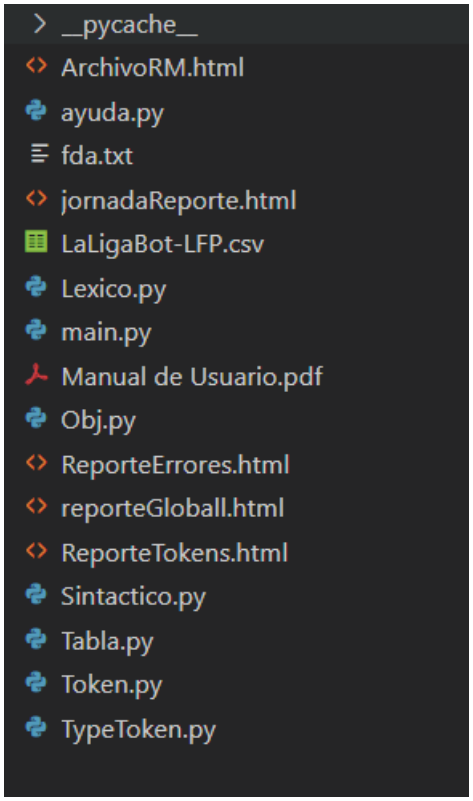
✓ REQUISITOS ADICIONALES

- Para el desarrollo de aplicaciones de la Tienda Windows y universales de Windows
  - El desarrollo de Windows 8.1 y Windows Phone 8.1 requiere Windows 8.1 Update o posterior.
  - El desarrollo de Windows Phone 8.0 requiere Windows 8.1 Update (x64) o posterior.
  - Para los emuladores de Windows, Windows 8.1 (x64) Professional Edition o versiones posteriores y un procesador que admita el Cliente Hyper-V y la traducción de direcciones de segundo nivel (SLAT).

II. Inicio.

Clases:

Para este programa se usaron 8 clases, las cuales son “main.py”, “Obj.py”, “ayuda.py”, “Lexico.py”, “Token.py”, “TypeToken.py”, ”Tabla.py”, “Sintactico.py” la función del main donde se encuentra la parte gráfica con ayuda de “ayuda.py” y se manda al analizador para guardar los datos. En léxico se encuentra el analizador léxico para guardar cada token y poder realizar los correspondientes html, de esta clase de manda a llamar a “Tabla.py”, “Obj.py” y “Token.py”. “Obj.py” y “Tabla.py” es mi clase de objetos. “ayuda.py” es para analizar el archivo csv. Y por último “Token.py” y “TypeToken.py” es donde se guarda cada token ingresado por el analizador.

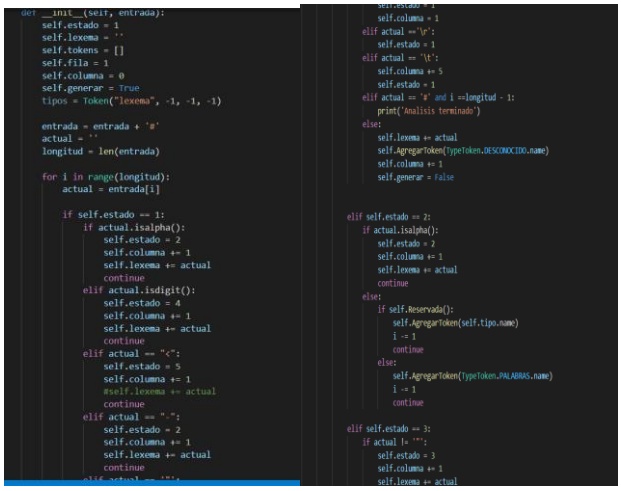


Clase Analizador (Léxico.py):

```
Lexico.py > Analizador > Reservada
from Token import Token
from TypeToken import TypeToken
from tkinter import messagebox
import webbrowser

class Analizador:
    tipo = TypeToken.DESCONOCIDO
    lexema = ''
    tokens= []
    estado = 1
    fila = 1
    columna = 0
    generar = False
    def __init__(self, entrada):···

    def AgregarToken(self,tipo):···

    def Reservada(self):···

    def ImprimirTo(self):···

    def ImprimirEr(self):···


    def guardarDatos(self):···
            #self.lista.append(lista(aux_tipo,aux_valor,"","","",aux_evento))
    def reporteTokens(self):···

    def reporteErorres(self):···
```

Método __init__():

En este método se encuentra toda la parte del analizador, por el cual va pasando por estados para poder guardar cada palabra, carácter o símbolo en su correspondiente token, esto con la ayuda de los métodos “agregarToken”, “reservada”.



Método AgregarToken():

Este método hace la función de guardar el lexema en el correspondiente token ingresado.

```
def AgregarToken(self,tipo):
    self.tokens.append(Token(self.lexema, tipo, self.fila, self.columna))
    self.lexema = ""
    self.estado = 1
    self.tipo = TypeToken.DESCONOCIDO
```

Método Resevarda():

Este método es el complemente del analizador para verificar si la palabra ingresada mediante el archivo es una palabra reservada o en otras palabras, si es una palabra establecida en mi lenguaje.

```
def Reservada(self):
    palabra = self.lexema.upper();
    #lista_palabras = ['RESULTADO', 'VS','TEMPORADA','JORNADA','GOLES',
    if palabra == 'RESULTADO':
        self.tipo = TypeToken.RESULTADO
        return True
    if palabra == 'VS':
        self.tipo = TypeToken.VS
        return True
    if palabra == 'TEMPORADA':
        self.tipo = TypeToken.TEMPORADA
        return True
    if palabra == 'JORNADA':
        self.tipo = TypeToken.JORNADA
        return True
    if palabra == 'GOLES':
        self.tipo = TypeToken.GOLES
        return True
    if palabra == 'LOCAL':
        self.tipo = TypeToken.C_GOLES
        return True
    if palabra == 'VISITANTE':
        self.tipo = TypeToken.C_GOLES
        return True
    if palabra == 'TOTAL':
        self.tipo = TypeToken.C_GOLES
        return True
    if palabra == 'TABLA':
        self.tipo = TypeToken.TABLA_TEM
        return True
    if palabra == 'PARTIDOS':
        self.tipo = TypeToken.PARTIDOS
        return True
    if palabra == 'TOP':
        self.tipo = TypeToken.TOP
    if palabra == 'TABLA':
        self.tipo = TypeToken.TABLA_TEM
        return True
    if palabra == 'PARTIDOS':
        self.tipo = TypeToken.PARTIDOS
        return True
    if palabra == 'TOP':
        self.tipo = TypeToken.TOP
        return True
    if palabra == 'SUPERIOR':
        self.tipo = TypeToken.C_TOP
        return True
    if palabra == 'INFERIOR':
        self.tipo = TypeToken.C_TOP
        return True
    if palabra == 'ADIOS':
        self.tipo = TypeToken.ADIOS
        return True
    if palabra == '-F':
        self.tipo = TypeToken.F
        return True
    if palabra == '-JI':
        self.tipo = TypeToken.JI
        return True
    if palabra == '-JF':
        self.tipo = TypeToken.JF
        return True
    return False
```

Método ImprimirTo():

Este método solo imprime los tokens, palabras, caracteres y símbolos ya establecidos en mi lenguaje.

```
def ImprimirTo(self):
    print("---Tokens---")
    tipos = Token("lexema", -1, -1, -1)
    for x in self.tokens:
        if str(x.tipo) != "DESCONOCIDO":
            print(x.lexema," --> ",str(x.tipo),' --> ',str(x.fila), ' --> ',str(x.columna))
```

Método ImprimirEr():

Este método solo imprime los tokens, palabras, caracteres y símbolos no establecidos en mi lenguaje.

```
def ImprimirEr(self):
    print("---TokensErrores---")
    tipos = Token("lexema", -1, -1, -1)
    for x in self.tokens:
        if str(x.tipo) == "DESCONOCIDO":
            print(str(x.lexema)," --> ",str(x.fila), ' --> ',str(x.columna),'--> Error Lexico')
```

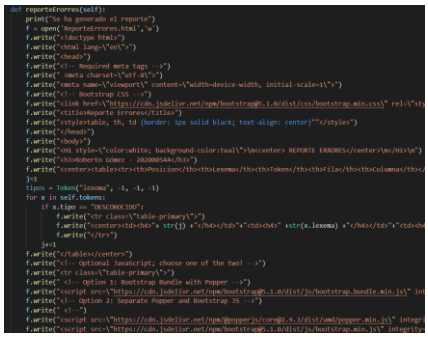
Método reporteTokens():

La función es crear el html con cada token y lexema ingresado al software. Todo se estará imprimiendo en una tabla.

```
print("Se ha generado el reporte")
f = open('ReporteTokens.html','w')
f.write("<!doctype html>")
f.write("<html lang=\"en\">")
f.write("<head>")
f.write("<!-- Required meta tags -->")
f.write(" <meta charset=\"utf-8\">")
f.write("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">")
f. (method) write: (__s: str, /) -> int
f.write("<link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css\" rel=\"stylesheet\" integrit
f.write("<title>Reporte Tokens</title>")
f.write("<style>table, th, td {border: 1px solid black; text-align: center}""</style>")
f.write("</head>")
f.write("<body>")
f.write("<H1 style=\"color:white; background-color:teal\">\n<center> REPORTE TOKENS</center>\n</H1>\n")
f.write("<h3>Roberto Gómez - 202000544</h3>")
f.write("<center><table><tr><th>Posicion</th><th>Lexema</th><th>Token</th><th>Fila</th><th>Columna</th></tr>")
j=1
tipos = Token("lexema", -1, -1, -1)
for x in self.tokens:
    if x.tipo != "DESCONOCIDO":
        f.write("<tr class=\"table-primary\">")
        f.write("<center><td><h4>"+ str(j) +"</h4></td>"+"<td><h4>" +str(x.lexema) +"</h4></td>"+"<td><h4>"+ str(x.tipo)+"<
        f.write("</tr>")
    j+=1
f.write("</table></center>")
f.write("<!-- Optional JavaScript; choose one of the two! -->")
f.write("<tr class=\"table-primary\">")
f.write(" <!-- Option 1: Bootstrap Bundle with Popper -->")
f.write("<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js\" integrity=\"sha384-U1
f.write("<!-- Option 2: Separate Popper and Bootstrap JS -->")
f.write(" <!--")
f.write("<script src=\"https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js\" integrity=\"sha384-eMNCOe
f.write("<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.min.js\" integrity=\"sha384-cn7l7gDp0
f.write("-->")
```

Método reporteErrores():

Así como en el método anterior su función es la misma, solo que en vez de tokens es tokens desconocidos o de error.



```
def reporteErrores(self):
    print("Se ha generado el reporte")
    f = open('ReporteErrores.html','w')
    f.write("<!doctype html>")
    f.write("<html lang=\"en\">")
    f.write("<head>")
    f.write("<!-- Required meta tags -->")
    f.write(" <meta charset=\"utf-8\">")
    f.write("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">")
    f.write("<!-- Bootstrap CSS -->")
    f.write("<link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css\" rel=\"st
    f.write("<title>Reporte Errores</title>")
    f.write("<style>table, th, td {border: 1px solid black; text-align: center}""</style>")
    f.write("</head>")
    f.write("<body>")
    f.write("<H1 style=\"color:white; background-color:teal\">\n<center> REPORTE ERRORES</center>\n</H1>\n")
    f.write("<h3>Roberto Gómez - 202000544</h3>")
    f.write("<center><table><tr><th>Posicion</th><th>Lexema</th><th>Token</th><th>Fila</th><th>Columna</th></
    j=1
    tipos = Token("lexema", -1, -1, -1)
    for x in self.tokens:
        if x.tipo == "DESCONOCIDO":
            f.write("<tr class=\"table-primary\">")
            f.write("<center><td><h4>"+ str(j) +"</h4></td>"+"<td><h4>" +str(x.lexema) +"</h4></td>"+"<td><h4
            f.write("</tr>")
        j+=1
    f.write("</table></center>")
    f.write("<!-- Optional JavaScript; choose one of the two! -->")
    f.write("<tr class=\"table-primary\">")
    f.write(" <!-- Option 1: Bootstrap Bundle with Popper -->")
    f.write("<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js\" int
    f.write("<!-- Option 2: Separate Popper and Bootstrap JS -->")
    f.write(" <!--")
    f.write("<script src=\"https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js\" integr
    f.write("<script src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.min.js\" integrity
```

Clase lista (Tabla.py):

```python
class Tablita:
    def __init__(self,equipo,puntos):
        self.equipo = equipo
        self.puntos = puntos


    def __repr__(self):
        return f'\n Equipo {self.equipo} Puntos {self.puntos} '

class Tablita_1:
    def __init__(self,equipo,puntos):
        self.equipo = equipo
        self.puntos = puntos


    def __repr__(self):
        return f'\n Equipo {self.equipo} Puntos {self.puntos} '
```

Clase lista (Obj.py):

```python
class Obj:
    def __init__(self,fecha, temporada, jornada, local, visitante,mar_l, mar_v):
        self.fecha = fecha
        self.temporada = temporada
        self.jornada = jornada
        self.local = local
        self.visitante = visitante
        self.mar_l = mar_l
        self.mar_v = mar_v

    def __repr__(self):
        return f'\n Fecha {self.fecha} Temporada {self.temporada} Jornada {self.jornada} Local {self.local} Visitante {self.visitante}
```

Clase TypeToken(TypeToken.py):

Esta clase es donde se encuentra todo mi lenguaje.

```python
from enum import Enum

class TypeToken(Enum):
    RESULTADO = 1
    VS = 2
    TEMPORADA = 3
    JORNADA = 4
    F = 5
    GOLES = 6
    G_LOCAL = 7
    G_VISITANTE = 8
    G_TOTAL = 9
    TABLA_TEM = 10
    PARTIDOS = 11
    JF = 12
    JI = 13
    TOP = 14
    N = 15
    SUP = 16
    INF = 17
    ADIOS = 18
    CADENA = 19
    NUMERO = 20
    FECHA = 21
    PALABRAS = 22
    DESCONOCIDO = 23
    C_TOP = 24
    C_GOLES = 25
    ULTIMO = 26
```

Clase (Token.py):

Aquí se encuentra el objeto del token.

```
class Token():
    def __init__(self,lexema,tipo,fila,columna):
        self.lexema = lexema
        self.tipo = tipo
        self.fila = fila
        self.columna = columna
```

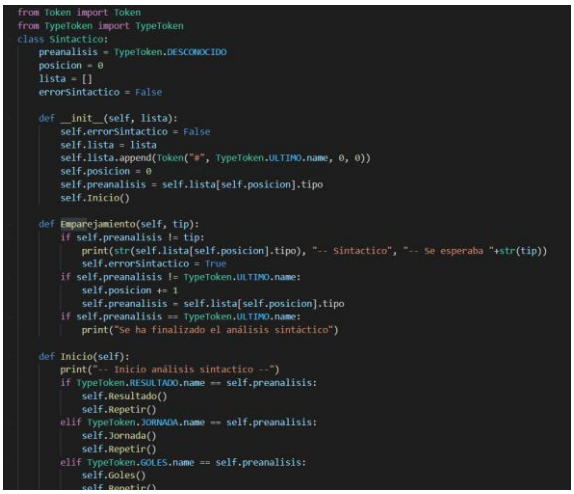
Clase (Help.py):

Esta su única función leer el archivo .csv

```
import csv
from Obj import Obj


def lectura():
    aux = []
    with open('LaLigaBot-LFP.csv') as a:
        reader = csv.reader(a)
        for aa in reader:
            aux.append(Obj(aa[0],aa[1],aa[2],aa[3],aa[4],aa[5],aa[6]))
    return aux
```

Clase (Help.py):

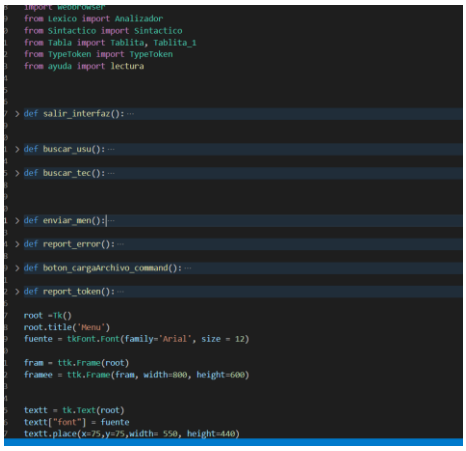
Aquí se encuentra el análisis sintáctico.



```
from Token import Token
from TypeToken import TypeToken
class Sintactico:
    preanalisis = TypeToken.DESCONOCIDO
    posicion = 0
    lista = []
    errorSintactico = False

    def __init__(self, lista):
        self.errorsintactico = False
        self.lista = lista
        self.lista.append(Token("#", TypeToken.ULTIMO.name, 0, 0))
        self.posicion = 0
        self.preanalisis = self.lista[self.posicion].tipo
        self.Inicio()

    def Emparejamiento(self, tip):
        if self.preanalisis != tip:
            print(str(self.lista[self.posicion].tipo), "-- Sintactico", "-- Se esperaba "+str(tip))
            self.errorSintactico = True
        if self.preanalisis != TypeToken.ULTIMO.name:
            self.posicion += 1
            self.preanalisis = self.lista[self.posicion].tipo
        if self.preanalisis == TypeToken.ULTIMO.name:
            print("Se ha finalizado el análisis sintáctico")

    def Inicio(self):
        print("-- Inicio análisis sintactico --")
        if TypeToken.RESULTADO.name == self.preanalisis:
            self.Resultado()
            self.Repetir()
        elif TypeToken.JORNADA.name == self.preanalisis:
            self.Jornada()
            self.Repetir()
        elif TypeToken.GOLES.name == self.preanalisis:
            self.Goles()
            self.Repetir()
```

Clase main.py:

Por ultimo se encuentra el main, en donde esta toda la parte grafica del software.



```
from Lexico import Analizador
from Sintactico import Sintactico
from Tabla import Tablita, Tablita_1
from TypeToken import TypeToken
from ayuda import lectura


def salir_interfaz():...


def buscar_usu():...

def buscar_tec():...


def enviar_men():...

def report_error():...

def boton_cargaArchivo_command():...

def report_token():...

root =Tk()
root.title('Menu')
fuente = tkFont.Font(family='Arial', size = 12)

fram = ttk.Frame(root)
framee = ttk.Frame(fram, width=800, height=600)


textt = tk.Text(root)
textt["font"] = fuente
textt.place(x=75,y=75,width= 550, height=440)
```

<Inicio>::= <Resultado> <Repetir>
| <Jornada> <Repetir>
| <Goles> <Repetir>
| <Tabla Temporada> <Repetir>
| <Partidos> <Repetir>
| <Top> <Repetir>
| <Adios> <Repetir>

<Repetir>::= <Resultado> <Repetir>
| <Jornada> <Repetir>
| <Goles> <Repetir>
| <Tabla Temporada> <Repetir>
| <Partidos> <Repetir>
| <Top> <Repetir>
| <Adios> <Repetir>
| Epsilon

<Resultado>::= tk_Resultado tk_Equipo tk_VS tk_Equipo tk_Temporada tk_Fecha
<Jornada>::= tk_Jornada tk_Numero tk_Temporada tk_Fecha tk_FArchivo tk_words
<Goles>::= tk_Goles <CondionGoles> tk_Equipo tk_Temporada tk_Fecha

<CondionGoles>::= tk_Local
| tk_Visitante
| tk_Total

<Tabla Temporada>::= tk_TablaTemporada tk_Temporada tk_Fecha tk_FArchivo tk_words
<Partidos>::= tk_Partidos tk_Equipo tk_Temporada tk_Fecha tk_FArchivo tk_words tk_Jl tk_Numero tK_JF tk_Numero
<Top>::= tk_Top <CondicionTop> tk_Temporada tk_Fecha tk_N tk_Numero

<CondicionTop>::= tk_Superior
| tk_Inferior

<Adios>::= tk_Adios