Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Ciencias y Sistemas

Lab. Lenguajes Formales y de Programación

MANUAL TECNICO

Justin Josue Aguirre Román

202004734

INTRODUCCION

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema, árboles y grafo de autómata utilizado para su desarrollo.

OBJETIVO DE ESTE MANUAL

El objetivo primordial de este Manual es ayudar y guiar al técnico a informarse y utilizar herramientas utilizadas en el software Reproductor MP3, para de esa manera poder hacer uso de la información deseada para poder despejar todas las dudas existentes y para poder comprender:

• Guía para gestión de herramientas para poner en funcionamiento el sistema de reproducción de canciones mp3.

• Conocer cómo utilizar el sistema, mediante una descripción detallada e ilustrada de las opciones.

• Conocer el alcance de toda la información por medio de una explicación detallada e ilustrada de cada una de las páginas que lo conforman el manual técnico.

REQUERIMIENTOS

El sistema pude ser instalado en cualquier sistema operativo que cumpla con los siguientes requerimientos:

a. Sistema Operativo: Cualquiera con una fecha de salida del 2014 en adelante.

a. Windows

i. Windows 10 (8u51 y superiores)

ii. Windows 8.x (escritorio)

iii. Windows 7 SP1

iv. Windows Vista SP2

v. Windows Server 2008 R2 SP1 (64 bits)

vi. Windows Server 2012 y 2012 R2 (64 bits)

vii. RAM: 128 MB

viii. Espacio en disco: 124 MB para JRE; 2 MB para Java Update

ix. Procesador: Mínimo Pentium 2 a 266 MHz

x. Exploradores: Internet Explorer 9 y superior, Firefox

b. Mac OS X

i. Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+

ii. Privilegios de administrador para la instalación

iii. Explorador de 64 bits

iv. Se requiere un explorador de 64 bits (Safari, por ejemplo) para ejecutar Oracle Java en Mac.

c. Linux

i. Oracle Linux 5.5+1

ii. Oracle Linux 6.x (32 bits), 6.x (64 bits)2

iii. Oracle Linux 7.x (64 bits)2 (8u20 y superiores)

iv. Red Hat Enterprise Linux 5.5+1 6.x (32 bits), 6.x (64 bits)2

v. Red Hat Enterprise Linux 7.x (64 bits)2 (8u20 y superiores)

vi. Suse Linux Enterprise Server 10 SP2+, 11.x

vii. Suse Linux Enterprise Server 12.x (64 bits)2 (8u31 y superiores)

viii. Ubuntu Linux 12.04 LTS, 13.x

ix. Ubuntu Linux 14.x (8u25 y superiores)

x. Ubuntu Linux 15.04 (8u45 y superiores)

xi. Ubuntu Linux 15.10 (8u65 y superiores)

DESCRIPCION DE ANALIZADORES Y GRAMATICA

Para la utilización de un autómata se debe contar con la creación de su representación gráfica (por medio de un grafo) y a su vez la creación de los métodos de árbol que generan el grafo y la delimitación de los Tokens y que tipo de Tokens son aceptados por el mismo.

Lista de Tokens Aceptados por el proceso de análisis léxico:

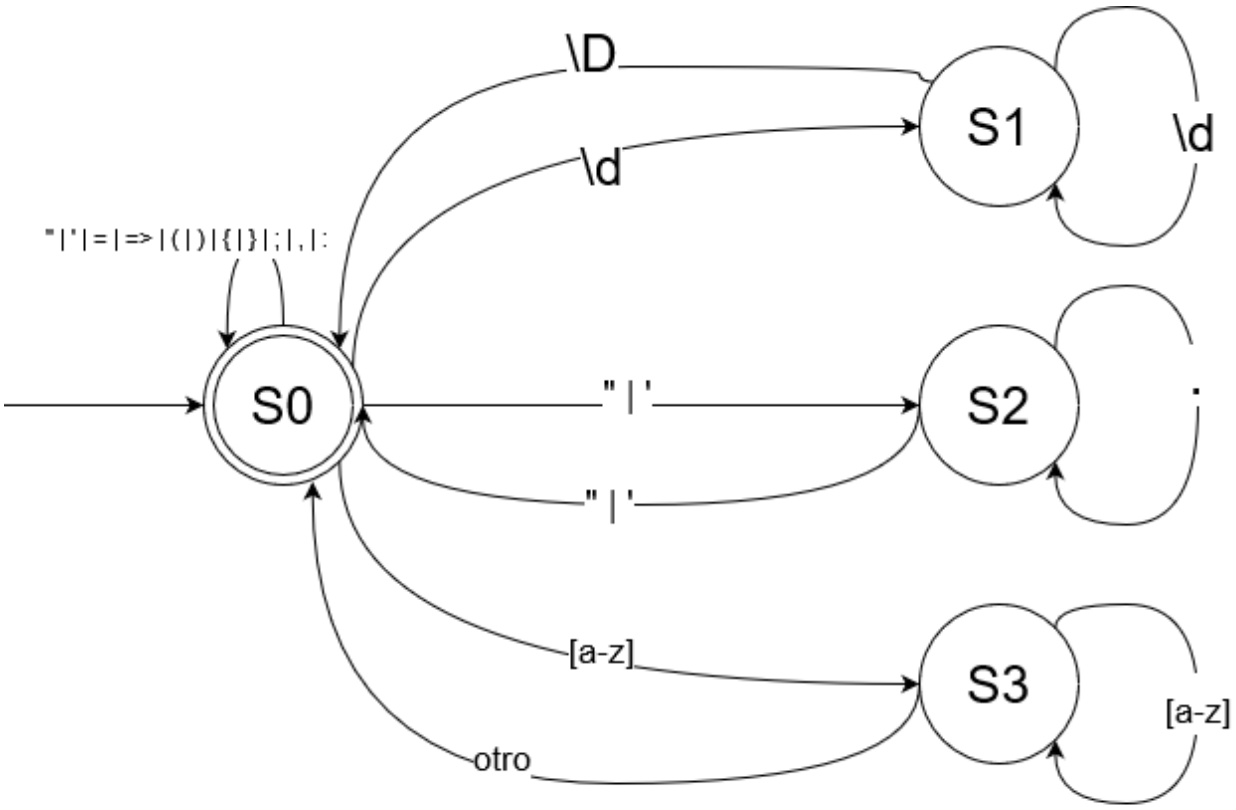
TOKEN	PATRON
REPLIST	Palabra replist
NOMBRE	Palabra nombre
ARTISTA	Palabra artista
RUTA	Palabra ruta
GENERO	Palabra genero
REPETIR	Palabra repetir
ANIO	Palabra anio
LLAVE_ABRE	Carácter llave que abre
LLAVE_CIERRE	Carácter llave que cierra
PARENTESIS_ABRE	Carácter paréntesis que abre
PARENTESIS_CIERRA	Carácter paréntesis que cierra
DOS_PUNTOS	Carácter dos puntos
FLECHA	Carácter corchete que cierra
IGUAL	Carácter igual
COMA	Carácter coma
PUNTO_COMA	Carácter punto y coma
CADENA	Inicia y termina con " o '

Tokens Validos, Elaboración propia 2021

DIAGRAMA AUTÓMATA ANÁLISIS LÉXICO

Se dividió el proceso de reconocimiento de los diferentes tokens con un análisis léxico que ayuda a identificar tanto las “palabras reservadas” como distintos símbolos que son aceptados por el lenguaje.

A continuación el diagrama del autómata:



METODOS UTILIZADOS

Para el desarrollo del software se dispuso de la creación de distintas variables globales para el intercambio de información entre los distintos métodos y de la importación de distintas librerías añadidas de forma por defecto en Python.

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMessageBox
from AnalizadorLexicoP import AnalizadorLexico
from AnalizadorSintactico import AnalizadorSintactico
import copy
import webbrowser
import os

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        self.Ruta=""
        self.scanner = AnalizadorLexico()
        self.Sintactico = AnalizadorSintactico()
```

Y la creación de distintos métodos para la lectura, análisis y generación de reportes que enlazaran y conectaran las distintas clases y archivos tipo py que conforman y hacen posible el funcionamiento del software.

Para la función cargar se utiliza la librería y herramientas QFileDialog, que guardara la ruta del archivo a analizar, esta ruta será utilizada para saber la ubicación del documento, abrirlo y obtener el contenido en texto, sin embargo, primero se verificara el tipo de extensión del archivo seleccionado, la cual obligatoriamente debe ser una extensión de tipo “mt3”.

```
    def CargarArchivo(self):
        buscar = QFileDialog.getOpenFileName()
        size=len(buscar[0])
        final=""
        self.ruta=buscar[0]
        for x in buscar[0]:
            if size<5:
                final+=x
            size-=1
        if final==".mt3":
            msg=QMessageBox()
            msg.setWindowTitle("SUCCES")
            msg.setText("Extension de archivo aceptado")
            msg.setIcon(QMessageBox.Information)
            x=msg.exec_()
            self.Analizar_BTN.setDisabled(False)
        else:
            msg=QMessageBox()
            msg.setWindowTitle("OCURRIO UN ERROR")
            msg.setText("Extension de archivo incorrecta")
            msg.setIcon(QMessageBox.Warning)
            x=msg.exec_()
```

La función para analizar el archivo llama los métodos de los analizadores léxico y sintáctico, para ejecutar sus funciones de inicio justamente en ese orden. Al completar mostrar un mensaje que según se haya completado o no la lectura y análisis de forma satisfactoria será el mensaje a mostrar en la interfaz gráfica.

```
def Analizar(self):
    self.scanner.listaErrores=[]
    self.scanner.listaTokens=[]
    self.Sintactico.listaErrores=[]
    self.Sintactico.ListaTokens=[]
    codigo_fuente = open(self.Ruta, 'r')
    contenido = codigo_fuente.read()
    codigo_fuente.close()
    self.scanner.analizar(contenido)
    self.scanner.impTokens()
    print("========================================================================")
    print("========================================================================")
    self.scanner.impErrores()
    cadena=self.Sintactico.analizar(self.scanner.listaTokens)
    self.Sintactico.ImpErrores()
    self.ConsolaArea.setText(cadena)
```

El método para los archivos de tipo HTML para los Reportes de Errores y Tokens usaran la misma base, las listas llenadas por medio de los análisis serán las llamadas en este apartado, recorriéndolas y guardándola en forma de una cadena de texto para llenar el formato de un archivo HTML.

```
def Reportes(self):
    Reporte=self.CmbReporte.currentText()
    contenido=""
    if Reporte=="ERRORES":
        c=0
        file=open('Errores.html','w')
        contenido="""<!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="styl
        <title>REPORTE ERRORES</title>
        </head>
        <body><center><h1>REPORTE DE ERRORES</h1></center><br>
        <div WIDTH='700'>
        <table class='table table-striped table-hover'  border='1'style='margin-left:auto;margin-right
        <tr><th>Lexema</th><th>Tipo</th><th>Linea</th><th>Columna</th></tr>"""
        for x in self.scanner.listaErrores:
            contenido+='<tr><td>'+str(x.descripcion)+'</td>'
            contenido+='<td>'+str(x.tipo)+'</td>'
            contenido+='<td>'+str(x.linea)+'</td>'
            contenido+='<td>'+str(x.columna)+'</td></tr>'
        for x in self.Sintactico.listaErrores:
            contenido+='<tr><td>'+str(x.descripcion)+'</td>'
            contenido+='<td>'+str(x.tipo)+'</td>'
            contenido+='<td>'+str(x.linea)+'</td>'
            contenido+='<td>'+str(x.columna)+'</td></tr>'
        contenido+='</tbody></table>'
        contenido+='</body></div></html>'
```

Por último la reproducción de archivos mp3 se exporto la librería mixer proveniente de pygame. Esta librería permite ejecutar exactamente las opciones de un reproductor de música (pausa, parar, reproducir). Y para las funciones restantes se gestiono con ayuda de objetos creados en base al archivo de entrada

```
Ui_MainWindow > AnalizarArchivo
while i<len(self.scanner.listaTokens):
    if estado==0:
        if self.scanner.listaTokens[i].tipo=="LlaveA":
            estado=1
    elif estado==1:
        if self.scanner.listaTokens[i].tipo=="NOMBRE":
            i+=2
            nombre=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="ARTISTA":
            i+=2
            artista=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="RUTA":
            i+=2
            rutaCancion=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="GENERO":
            i+=2
            genero=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="REPETIR":
            i+=2
            repetir=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="ANIO":
            i+=2
            anio=self.scanner.listaTokens[i].lexema

        elif self.scanner.listaTokens[i].tipo=="LlaveC":
            Song=Cancion(nombre,artista,anio,repetir,rutaCancion,genero)
            print(Song.nombre)
            self.Canciones.append(Song)
```

Creación de Objetos.

```
def PlayMusica(self):
    if self.Stop_BTN.isEnabled():
        mixer.music.unpause()
        self.Pause_BTN.setDisabled(False)
        self.Play_BTN.setDisabled(True)
    else:
        self.idMusica=0
        ruta=self.Canciones[self.idMusica].rutaCancion+""
        mixer.init()
        mixer.music.load(ruta)
        mixer.music.play()
        info="INFORMACION\nNOMBRE:\t"+self.Canciones[self.idMusica].nombre+"\nARTISTA:\t"+self.Canciones[sel
        self.informacion_Area.setText(info)
        self.Play_BTN.setDisabled(True)
        self.Stop_BTN.setDisabled(False)
        self.Prev_BTN.setDisabled(False)
        self.Next_BTN.setDisabled(False)
        self.Pause_BTN.setDisabled(False)


def StopMusica(self):
    mixer.music.stop()
    self.Stop_BTN.setDisabled(True)
    self.Prev_BTN.setDisabled(True)
    self.Next_BTN.setDisabled(True)
    self.Pause_BTN.setDisabled(True)
    self.Play_BTN.setDisabled(False)
```

Uso de la librería mixer para creación de reproductor mp3.