Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Lenguajes Formales y de Programación

MANUAL TÉCNICO - PROYECTO 2

Kevin José de la Cruz Girón

Carnet: 202010844

Guatemala, Diciembre 2022

ÍNDICE

INTRODUCCIÓN

Este manual técnico tiene como finalidad dar a conocer al lector que pueda requerir hacer modificaciones futuras al software el desarrollo de la aplicación indicando el IDE utilizado para su creación, su versión, requerimientos del sistema, etc.

La aplicación tiene como objetivo brindarle al usuario los datos que necesite mediante comandos, el cual hará cargas de los archivos .glc y .ap donde se contiene toda la información, estos archivo contendrán parámetros que serán procesados por la aplicación y mostrados al usuario de forma de gramáticas y en forma de autómatas.

OBJETIVOS

General

Brindar al lector una guía que contenga la información del manejo de clases, atributos, métodos y del desarrollo de la aplicación desarrollada en el lenguaje de programación Python para facilitar futuras actualizaciones y modificaciones realizadas por terceros.

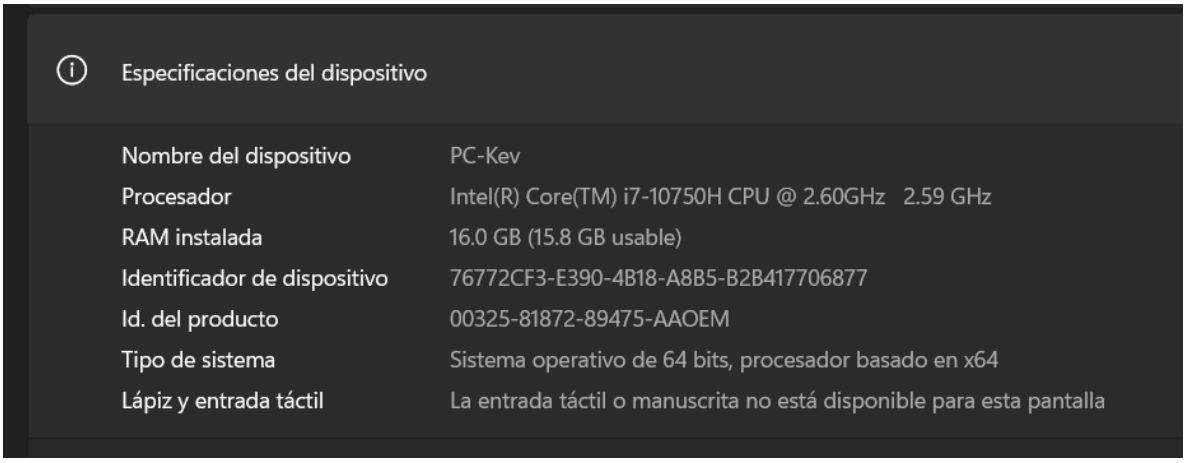
Específicos

- Proporcionar al lector una idea más precisa de los métodos y clases creadas para el desarrollo de la aplicación.
- Dar más información al lector de las herramientas utilizadas para el desarrollo de la aplicación.

REQUERIMIENTOS

- SO:

  Las especificaciones del equipo utilizado para el desarrollo de la práctica se verán reflejadas en la siguiente imagen:



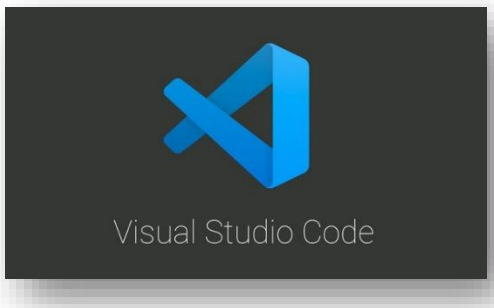
- PYTHON:

  Para realizar este proyecto se recomienda utilizar una versión mayor a la 3.8, en ese cao se utilizó la siguiente:

```
C:\Users\kevin>Python -V
Python 3.11.0
```

- IDE:

  Para el desarrollo del proyecto se trabajó con Visual Studio Code, queda a su discreción el ide a utilizar, sin embargo, por su fácil uso se recomienda.



Técnica de Programación

Programación orientada a objetos:

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientad a objetos disminuye los errores y promociona la reutilización del código, Python es un lenguaje orientado a objetos, por este motivo se decidió trabajar de este modo.

Convenciones de Nomenclatura

Declaración de Variables:

Se declaran variables que serán utilizados en la aplicación, en las cuales se incluyen listas, componentes GUI y todo lo necesario a lo que se deba acceder de manera global, para posteriormente se utilizado en los métodos, funciones y cálculos.

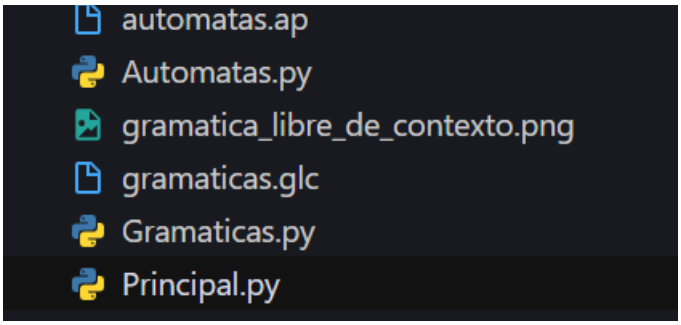
```
class PantallaPrincipal():
    def __init__(self):

        self.lista_nombres_Gramatica=[]
        self.lista_nombres_Automatas=[]
        self.lista_Gramaticas=[]
        self.lista_Automatas=[]
        self.lista_no_terminales=[]
        #self.moduloMain = ModuloPrincipal()
        self.ventana = Tk()
        self.ventana.resizable(True, False)  # Redimensionar la ventana
        self.ventana.title('Pantalla Principal')  # Titulo de la ventana
        self.ventana.geometry('1100x700')  # Tamaño de la ventana
        self.Centrar(self.ventana, 1000, 700)  # Centrar la ventana
        self.ventana.config(bg='#172a39')
        self.Ventana()  # Llamar a la ventana
        #self.a= ModuloPrincipal()
```

Los nombres de las variables son específicas para el uso y lo que almacena.

Clases:

Se declaran clases con el método UpperCamelCase, método el cual las palabras en su inicio son mayúscula , cada clase hace alusión a lo que hace y en lo que aporta al programa.



Modelos:

Se declaran los módelos con el método UpperCamelCase, método el cual las palabras en su inicio son mayúscula , cada modelo aporta al desarrollo con POO.

Modelo de Autómatas

```python
class Automatas:
    def __init__(self, nombre, alfabeto,simbolosPila,estados,estadoInicial,estadosAceptacion,transicio
        self.nombre = nombre
        self.alfabeto = alfabeto
        self.simbolosPila = simbolosPila
        self.estados = estados
        self.estadoInicial = estadoInicial
        self.estadosAceptacion = estadosAceptacion
        self.transiciones = transiciones


class Transiciones:
    def __init__(self,estadoOrigen,simboloEntrada,simboloextraPila,estadoDestino,simboloinsertaPila):
        self.estadoOrigen = estadoOrigen
        self.simboloEntrada = simboloEntrada
        self.simboloextraPila = simboloextraPila
        self.estadoDestino = estadoDestino
        self.simboloinsertaPila = simboloinsertaPila
```

Modelo de gramáticas

```
class Producciones:
    def __init__(self, origen, destinos):
        self.origen = origen
        self.destinos = destinos




class Gramatica:

    def __init__(self, nombre, no_terminales, terminales, nti, producciones):
        self.nombre = nombre
        self.no_terminales = no_terminales
        self.terminales = terminales
        self.nti = nti
        self.producciones = producciones
```

Métodos y funciones:

Se declaran métodos y funciones con el método LowerCamelCase, método el cual las palabras en su inicio son mayúsculas a excepción de la primera letra que debe ser minúscula, cada clase hace alusión a lo que hace y en lo que aporta al programa.

```
63
64          self.frame.mainloop()
65
66      def modGramatica(self):···
90
91
92      def mostrarInformacion(self):···
179
180
181     def Arbol_Gramatica(self):···
270
271
272     def modAutomata(self):
273         ventana = tk.Tk()
274         ventana.geometry('1000x600')
275         ventana.config(bg='#172a39')
276         ventana.title("Ventana Autómatas")
277         ventana.resizable(True, False)
278
279         Label(ventana, text="Módulo Autómata ", font=(
280             'Times New Roman', 40), fg='#ffffff', bg='#3d5568', width=40).grid(row=0,column=0)
281
```

Métodos Principales

Método ventanaArchivo():

En este método se abre la misma ventana GUI de carga para seleccionar el archivo a cargar, es la misma ventana tanto para la carga de gramáticas como la carga de autómatas.

```
def ventanaArchivo(self):
    ventanaCarga = tk.Tk()

    def almacenar(contenido):···

    def almacenarAutomata(contenido):

        x = 1
        lista_transiciones = []
        lista_alfabeto = []
        lista_simbolosPila = []
        lista_estados = []
        lctxt = False
        for file in contenido:
            file = file.rstrip('\n')
            if x==1:
                nombre=file
                if (nombre in self.lista_nombres_Automatas):
                    MessageBox.showwarning("Alerta", "El automata ***{nombre}***sera saltado debid
                    x = '%'
```

Método de módulos

En estos métodos se almacenan los módulos principales, los cuales con modAutomata y modGramatica los cuales son los principales en la aplicación respecto a la interfaz gráfica.

```
def modGramatica(self):
    ventana = tk.Tk()
    ventana.geometry('1000x600')
    ventana.config(bg='#172a39')
    ventana.title("Ventana Gramática")
    ventana.resizable(True, False)




    Label(ventana, text="Módulo Gramática ", font=(
        'Times New Roman', 40), fg='#ffffff', bg='#3d5568', width=40).grid(row=0,column=0)

    Button(ventana, text="Carga Archivos", command=self.ventanaArchivo, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=20).grid(row=2,column=0,pady=40)

    Button(ventana, text="Mostrar información General", command=self.mostrarInformacion, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=40).grid(row=3,column=0,pady=30)

    Button(ventana, text="Árbol de derivación", command=self.Arbol_Gramatica, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=20).grid(row=4,column=0,pady=30)
```

```python
def modAutomata(self):
    ventana = tk.Tk()
    ventana.geometry('1000x600')
    ventana.config(bg='#172a39')
    ventana.title("Ventana Autómatas")
    ventana.resizable(True, False)

    Label(ventana, text="Módulo Autómata ", font=(
        'Times New Roman', 40), fg='#ffffff', bg='#3d5568', width=40).grid(row=0,column=0)

    Button(ventana, text="Carga Archivos", command=self.ventanaArchivo, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=20).grid(row=2,column=0,pady=40)

    Button(ventana, text="Mostrar información de Autómata", command=self.mostrarInfoAuto, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=40).grid(row=3,column=0,pady=30)

    Button(ventana, text="Validar una cadena", command=self.validarCadena, font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=20).grid(row=4,column=0,pady=30)

    Button(ventana, text="Ruta de validación", command="", font=(
        'Times New Roman', 15), fg='#000000', bg='#a9c2d6', width=20).grid(row=5,column=0,pady=30)

    """Button(ventana, text="Recorrido paso a paso", command="", font=(
```

LIBRERÍAS

- import os:

  El módulo os en Python proporciona y expone los detalles y la funcionalidad del sistema operativo.

- from tkinter import filedialog, Tk:

  Librería útil para poder programar la interfaz de usuario.

- Import Graph:

  Librería encargada de generar los gráficos para los reportes en pdf.