Manual Técnico

Para los menús se utilizó la librería Tkinter y se creó una clase para cada pestaña del proyecto, para las características de las ventanas se utilizó lo siguiente:

```
def ventanaPrincipal():
    global ventanaPrincipal
    ventanaPrincipal = Tk()
    ventanaPrincipal.title("Menu Principal")

    screen_width = ventanaPrincipal.winfo_screenwidth()
    screen_height = ventanaPrincipal.winfo_screenheight()

    ventanaPrincipal_width = 300
    ventanaPrincipal_height = 300

    x = (screen_width - ventanaPrincipal_width) // 2
    y = (screen_height - ventanaPrincipal_height) // 2

    ventanaPrincipal.geometry(f"{ventanaPrincipal_width}x{ventanaPrincipal_height}+{x}+{y}")
    ventanaPrincipal.configure(bg="#37474f")

    Button(ventanaPrincipal, text="Gramatica libre de contexto", height="2", width="30", command = ventanaOFC).pack(padx=10, pady=10)
    Button(ventanaPrincipal, text="Automotas de pila", height="2", width="30", command = ventanaAP).pack(padx=10, pady=40)
    Button(ventanaPrincipal, text="Salir", height="2", width="30", command = salir).pack(padx=10, pady=40)

    ventanaPrincipal.mainloop()
```

Para las tablas que muestran los archivos cargados se utilizó Tkinter también

```
table = ttk.Treeview(ventanaInfoGeneral)
table.pack(padx=10, pady=10)
table['columns'] = ('glc_nombre', 'glc_noTerminal', 'glc_terminal', 'glc_inicial', 'glc_produccion')

table.column('#0', width=0, stretch=NO)
table.column('glc_nombre', anchor=CENTER, width=75)
table.column('glc_noTerminal', anchor=CENTER, width=100)
table.column('glc_terminal', anchor=CENTER, width=75)
table.column('glc_inicial', anchor=CENTER, width=50)
table.column('glc_produccion', anchor=CENTER, width=400)

table.heading('#0', text='', anchor=CENTER)
table.heading('glc_nombre', text='Nombre', anchor=CENTER)
table.heading('glc_noTerminal', text='No Terminales', anchor=CENTER)
table.heading('glc_terminal', text='Terminales', anchor=CENTER)
table.heading('glc_inicial', text='Inicial', anchor=CENTER)
table.heading('glc_produccion', text='Producciones', anchor=CENTER)
```

Y se utilizó un for para poder mostrar la información en esta tabla

```
cont = 0
for glc in glc_array:
    table.insert("", "end", iid=cont, text='',
    values=(glc.nombre, glc.noTerminal, glc.terminal, glc.inicial, glc.produccion))
    cont += 1
```

Para la carga de archivos llame al lector que se creó y al procesador para poder guardar la información proporcionada en los archivos seleccionados

```
def upload_file(case, glc_array, ap_array):

    type_file = ''
    # Lectura del archivo
    if (case == 1):
        type_file = '.glc'
    elif (case == 2):
        type_file = '.ap'

    data = filereader.file_reader(type_file)
    if data is None:
        print("- Ningun archivo seleccionado \n")
    else:
        processor.data_processor(data, case, glc_array, ap_array)


        if (data is None):
            messagebox.showinfo('Mensaje', 'Error al cargar la información.')
        else:
            messagebox.showinfo('Mensaje', 'Información cargada exitosamente.')
```

Para el procesador utilice un for para identificar si tenía letras, a la hora de detectar el salto de línea guardaba la información en una variable y pasaba a la siguiente línea a hacer lo mismo

```
# Init of processor
if case == 1:
    for letter in data:
        if f_automataGL == False:
            if letter != '\n':
                nombre_valorAutomataGL += letter
            else:
                f_automataGL = True

        elif f_noTerminal == False:
            if letter != '\n':
                if letter != ",":
                    noTerminal_valor += letter
                else:
                    noTerminal_list.append(noTerminal_valor)
                    noTerminal_valor = ""
```

Para guardar los Autómatas de Pila y las Gramáticas Libre de Contexto se creó una clase con sus características

```python
class apModel:

    def __init__(self, nombre, alfabeto, simbolo, estado, eInicial, eAceptacion, transicion):
        self.nombre = nombre
        self.alfabeto = alfabeto
        self.simbolo = simbolo
        self.estado = estado
        self.eInicial = eInicial
        self.eAceptacion = eAceptacion
        self.transicion = transicion
```