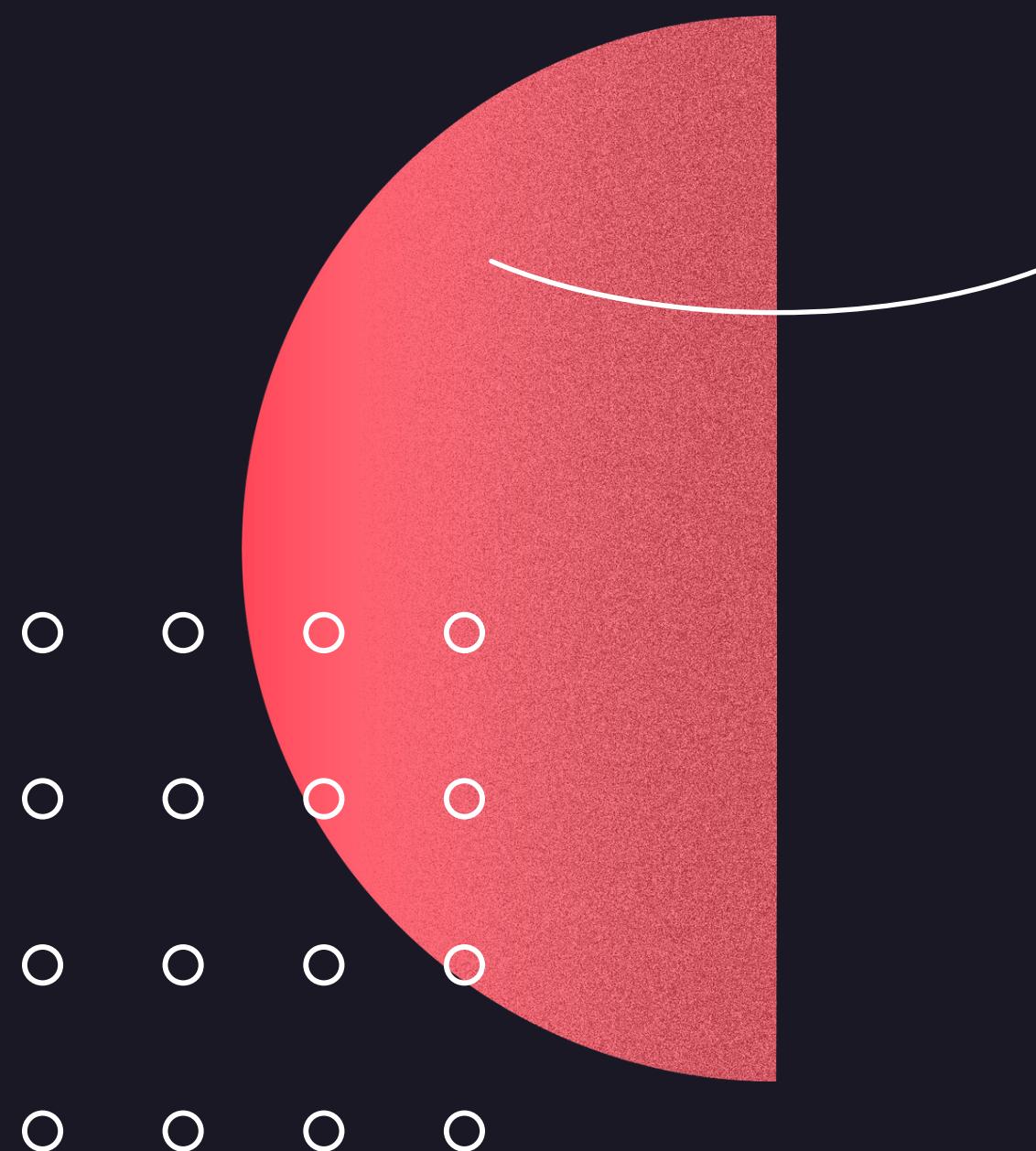




Manual Técnico

Steven Josue González Monroy

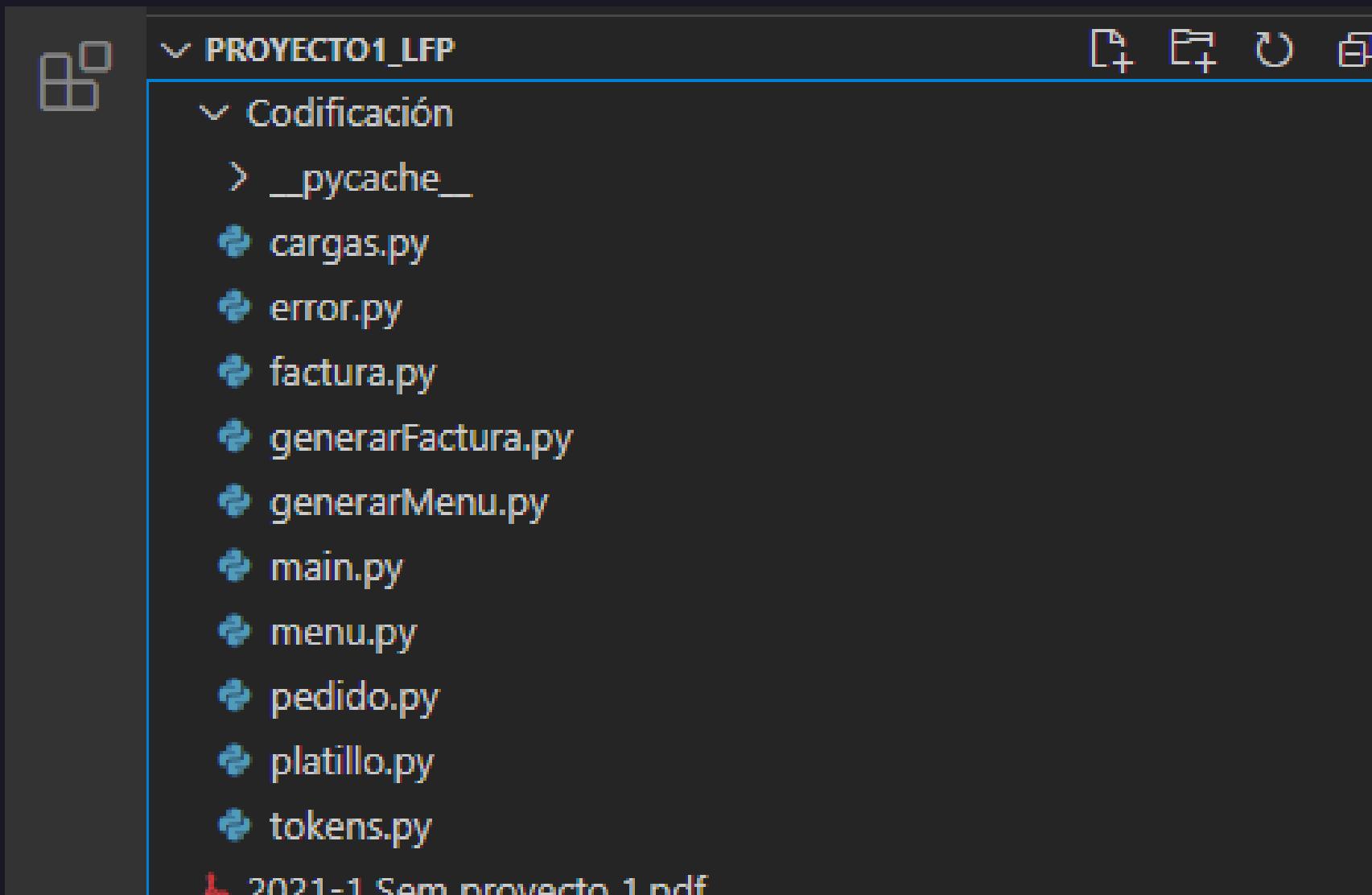
INTRODUCCIÓN



El siguiente manual guiará a los usuarios que harán soporte al sistema, el cual les dará a conocer los requerimientos y la estructura para la construcción del sistema, las herramientas necesarias para la construcción y la funcionalidad del sistema, así como los paradigmas y automatas utilizados.

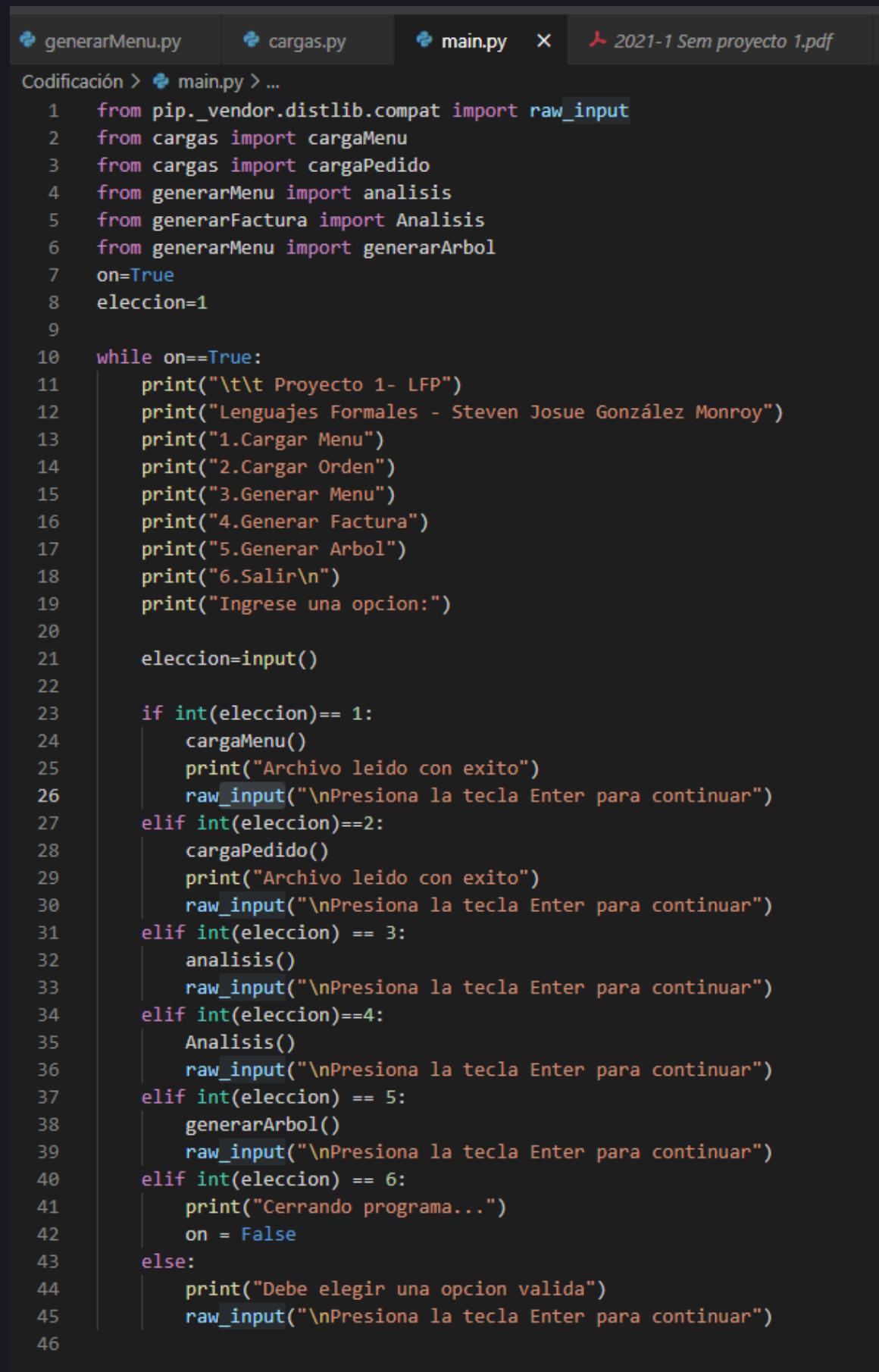
CODIFICACION:

El Proyecto 1 consta de las siguientes clases:



1.Main

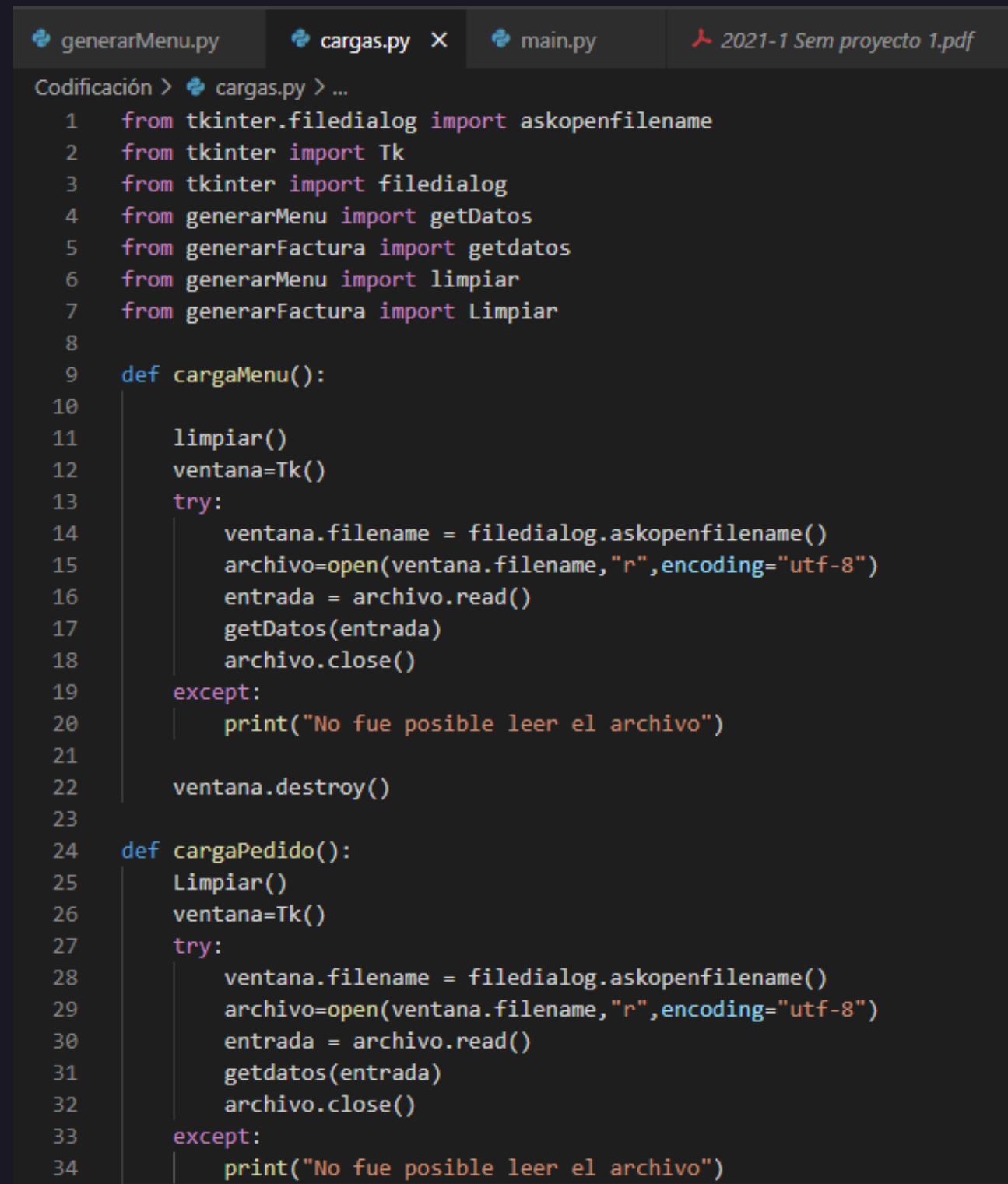
Este archivo contiene el código que inicializa el programa, contiene un paradigma estructurado ya que sus instrucciones se ejecutan una por una. Su objetivo es mostrar el menú principal con métodos print y hacer llamadas a los métodos que el usuario desee por medio de input.



```
generarMenu.py cargas.py main.py 2021-1 Sem proyecto 1.pdf
Codificación > main.py > ...
1  from pip._vendor.distlib.compat import raw_input
2  from cargas import cargaMenu
3  from cargas import cargaPedido
4  from generarMenu import analisis
5  from generarFactura import Analisis
6  from generarMenu import generarArbol
7  on=True
8  eleccion=1
9
10 while on==True:
11     print("\t\t Proyecto 1- LFP")
12     print("Lenguajes Formales - Steven Josue González Monroy")
13     print("1.Cargar Menu")
14     print("2.Cargar Orden")
15     print("3.Generar Menu")
16     print("4.Generar Factura")
17     print("5.Generar Arbol")
18     print("6.Salir\n")
19     print("Ingrese una opcion:")
20
21     eleccion=input()
22
23     if int(eleccion)== 1:
24         cargaMenu()
25         print("Archivo leido con exito")
26         raw_input("\nPresiona la tecla Enter para continuar")
27     elif int(eleccion)==2:
28         cargaPedido()
29         print("Archivo leido con exito")
30         raw_input("\nPresiona la tecla Enter para continuar")
31     elif int(eleccion) == 3:
32         analisis()
33         raw_input("\nPresiona la tecla Enter para continuar")
34     elif int(eleccion)==4:
35         Analisis()
36         raw_input("\nPresiona la tecla Enter para continuar")
37     elif int(eleccion) == 5:
38         generarArbol()
39         raw_input("\nPresiona la tecla Enter para continuar")
40     elif int(eleccion) == 6:
41         print("Cerrando programa...")
42         on = False
43     else:
44         print("Debe elegir una opcion valida")
45         raw_input("\nPresiona la tecla Enter para continuar")
```

2.Cargas

Esta clase contiene los métodos que obtienen y leen los archivos de entrada para su posterior análisis, se crea un cuadro de diálogo donde el usuario elige la ruta del archivo deseado, luego se guardan los datos en una cadena y la envían a otra clase para su análisis.



The screenshot shows a code editor interface with several tabs at the top: 'generarMenu.py', 'cargas.py' (which is currently active), and 'main.py'. A status bar at the bottom right indicates '2021-1 Sem proyecto 1.pdf'. The code in 'cargas.py' is as follows:

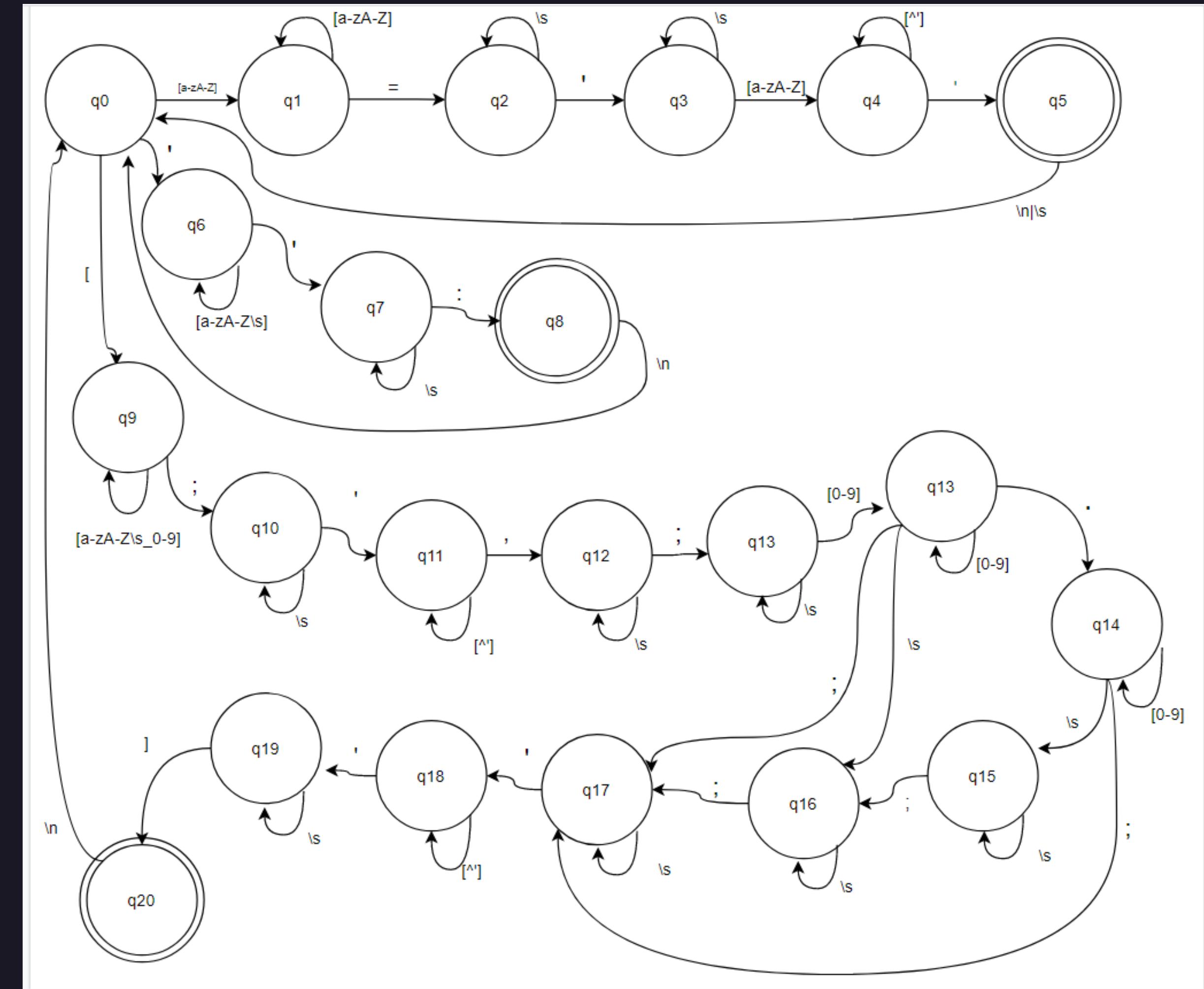
```
codificación > cargas.py > ...
1  from tkinter import askopenfilename
2  from tkinter import Tk
3  from tkinter import filedialog
4  from generarMenu import getDatos
5  from generarFactura import getdatos
6  from generarMenu import limpiar
7  from generarFactura import Limpiar
8
9  def cargaMenu():
10
11     limpiar()
12     ventana=Tk()
13     try:
14         ventana.filename = filedialog.askopenfilename()
15         archivo=open(ventana.filename,"r",encoding="utf-8")
16         entrada = archivo.read()
17         getDatos(entrada)
18         archivo.close()
19     except:
20         print("No fue posible leer el archivo")
21
22     ventana.destroy()
23
24  def cargaPedido():
25      Limpiar()
26      ventana=Tk()
27      try:
28          ventana.filename = filedialog.askopenfilename()
29          archivo=open(ventana.filename,"r",encoding="utf-8")
30          entrada = archivo.read()
31          getdatos(entrada)
32          archivo.close()
33      except:
34          print("No fue posible leer el archivo")
35      ventana.destroy()
```

3.generarMenu

Esta clase contiene el metodo que realiza el analisis de los datos obtenidos del archivo, por medio de un automata se determina caracter por caracter la estructura del menu, una vez finalizado el analisis se procede a generar un html concatenando cadenas de datos con la estructura y los datos obtenidos del analisis.

Si el análisis identifica errores, únicamente se generará una tabla de errores en vez del menú.

Automata para Menu:



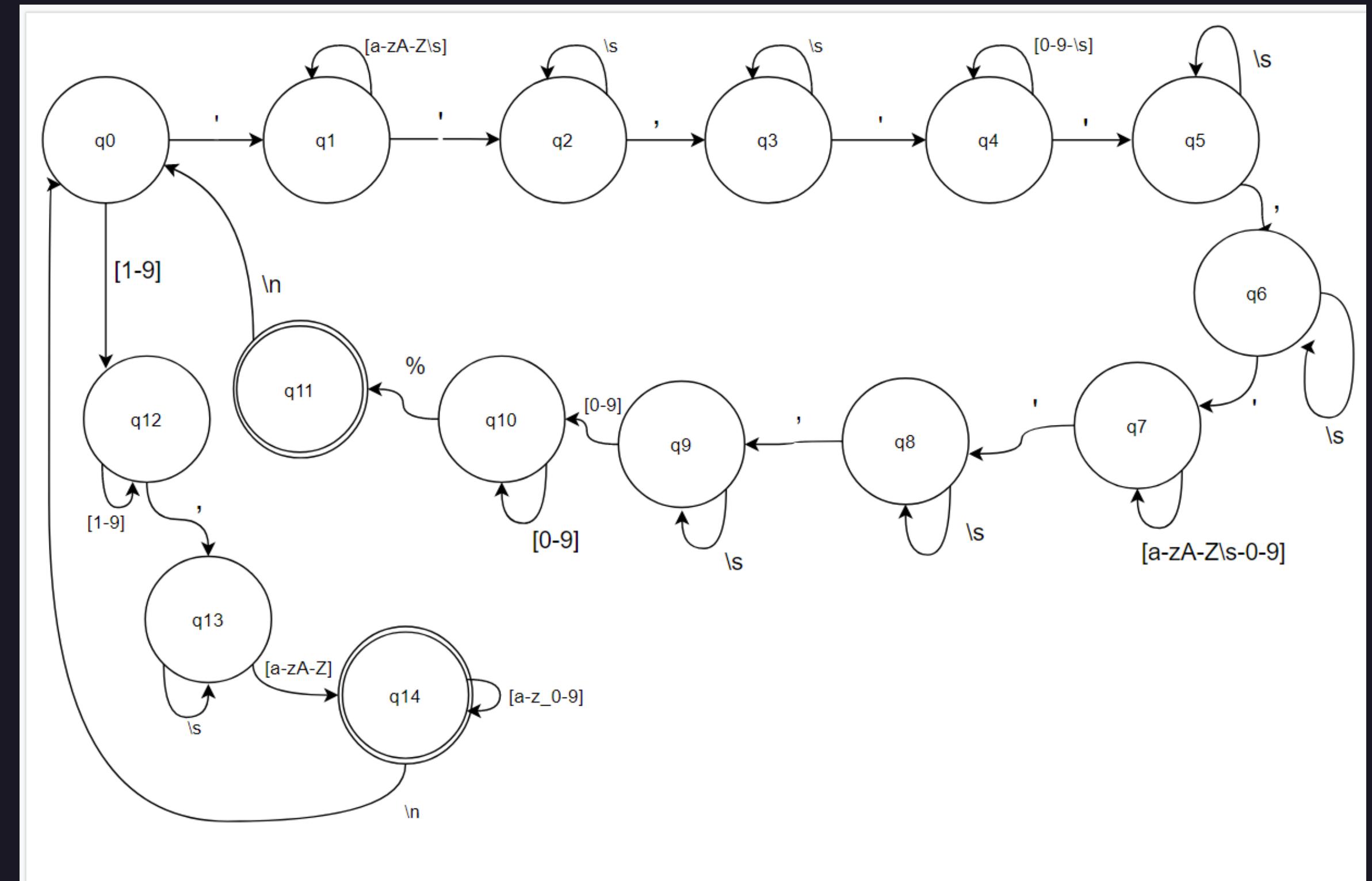
4.generarFactura

Esta clase contiene el metodo que analiza el archivo de entrada con los datos del pedido, recibe la cadena y por medio de un automata se determina si su estructura es correcta y se guarda la informacion, una vez finalizado el analisis se procede a generar un archivo html concatenando cadenas con la estructura del documento y los datos obtenidos del analisis. Si el archivo contiene errores generará una tabla de errores

```
icación > generadorFactura.py > Analisis
def generarFacturaHTML(factura):
    inicio="<!doctype html><html><head> <title>Factura</title> </head><body><h1 style=\"text-align: center;\">Factura No."+str(factura.no)+"</h1>
    <h2 style=\"text-align: center;\">Fecha: "+factura.fecha+"</h2><p style=\"text-align: center;\">&ampnbsp</p>
    <p style=\"text-align: left; padding-left: 600px;\"><strong>Datos del Cliente:</strong></p><p style=\"text-align: left; padding-left: 600px;\"><strong>NIT:</strong> "+factura.nit+"</p><p style=\"text-align: left; padding-left: 600px;\"><strong>Descripción:</strong></p>
    <table style="border-collapse: collapse; border-color: white; margin-left: auto; margin-right: auto; height: 95px; width: 495px;"><tbody><tr style="width: 191px; height: 23px; border-style: solid; text-align: center;"><td style="width: 101px; height: 18px; border-color: black; border-style: solid; text-align: center; vertical-align: middle;"><strong>Concepto&ampnbsp</strong></td><td style="width: 101px; height: 18px; border-color: black; border-style: solid; text-align: center; vertical-align: middle;"><strong>Cantidad</strong></td>
    mitad=""
    for element in factura_pedidos:
        mitad+=mitad+"|
|  |

```

Automata para Pedido:



5. Clases de objetos

Adicional a las clases anteriores para poder generar el menu y la factura se hizo uso del paradigma Orientado a Objetos por lo tanto se codificaron varias clases para guardar los datos de manera ordenada, estas contienen únicamente un constructor con los atributos que se reciben y los métodos self de cada atributo

```
menu.py - Proyecto1_LFP - Visual Studio Code
generarMenu.py  menu.py X  cargas.py  main.py
Codificación > menu.py > Menu > __init__
1 class Menu:
2     def __init__(self, restaurante, categorias, platillos):
3         self.restaurante=restaurante
4         self.categorias=categorias
5         self.platillos=platillos

Codificación > factura.py > ...
1 class Factura:
2     def __init__(self, cliente, nit, dirección, pedidos, total, propina, porcentaje, fecha, no, subtotal):
3         self.cliente=cliente
4         self.nit=nit
5         self.dirección=dirección
6         self.pedidos=pedidos
7         self.total=total
8         self.propina=propina
9         self.porcentaje=porcentaje
10        self.fecha=fecha
11        self.no=no
12        self.subtotal=subtotal

Codificación > pedido.py > ...
1 class Pedido:
2     def __init__(self, cantidad, identificador, nombre, costo, precio):
3         self.cantidad=cantidad
4         self.identificador=identificador
5         self.costo=costo
6         self.nombre=nombre
7         self.precio=precio

Codificación > Error.py > ...
1 class Error:
2     def __init__(self, numero, lexema, fila, columna, descripción):
3         self.numero=numero
4         self.lexema=lexema
5         self.fila=fila
6         self.columna=columna
7         self.caracter=caracter
8         self.descripción=descripción
```

Paradigmas

1. Paradigma Orientado a Objetos:

Este paradigma fue muy util para la realizacion del menu y la factura, ya que de esta manera se pueden guardar los datos de los archivos de entrada en objetos y asi ser almacenados de manera ordenada, facilitando su manipulacion posterior.

2. Paradigma funcional:

Este es el paradigma mas utilizado en el proyecto, ya que se utilizan funciones en la mayoria de clases esto permite que el flujo del programa no sea lineal y se puedan llamar instrucciones desde cualquier clase.

3. Paradigma descriptivo:

Este paradigma fue utilizado en menor medida pero esta presente en el uso de algunos metodos en las listas y en la generacion del grafico de abol como por ejemplo los metodos append(), source(), open(), view(), ya que no sabemos que instrucciones ejecutan estos metodos pero realizan la tarea deseada

Recomendaciones:

- Instalar graphviz en el path para que se ejecute de manera correcta
- Tener la version de pip siempre actualizada
- Verificar que los archivos de entrada esten correctamente estructurados