

GMFR

Generador de Menús y Facturas para Restaurantes

MANUAL TÉCNICO

Contenido

INTRODUCCIÓN:.....	3
REQUERIMIENTOS DEL SISTEMA.....	4
Windows:	4
Mac OS X:	4
Linux:	4
Menú	5
Cargar menú y orden:.....	6
Generar menú y orden:	7
Generar tablas de tokens/errores:	8
PARADIGMAS DE PROGRAMACIÓN UTILIZADAS	9
Programación estructurada	10
Programación procedimental.....	11
Programación modular.....	12
Programación Orientada a Objetos.....	12
CONCLUSIÓN:	15

INTRODUCCIÓN:

Se realizó una aplicación, que permite analizar y procesar información de un archivo de texto dado con un formato, el cual contiene una estructura y en base a esa estructura se identifican los diferentes patrones para extraer la información que contiene y con eso se generan reportes, se hizo a través del lenguaje Python, se hizo uso de HTML y CSS para generar reportes web y también se hizo uso de librerías externas como Tkinter para mostrar un explorador de archivos.

REQUERIMIENTOS DEL SISTEMA

Windows:

- Python 3.8.1
- Windows 10 (8u51 y superiores)
- Windows 8.x (escritorio)
- Windows 7 SP1 • Windows Vista SP2

Mac OS X:

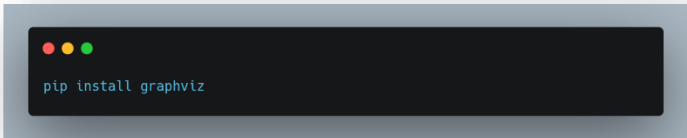
- Python 3.8.1
- Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+
- Privilegios de administrador para la instalación

Linux:

- Python 3.8.1
- Cualquier distribución de Linux.

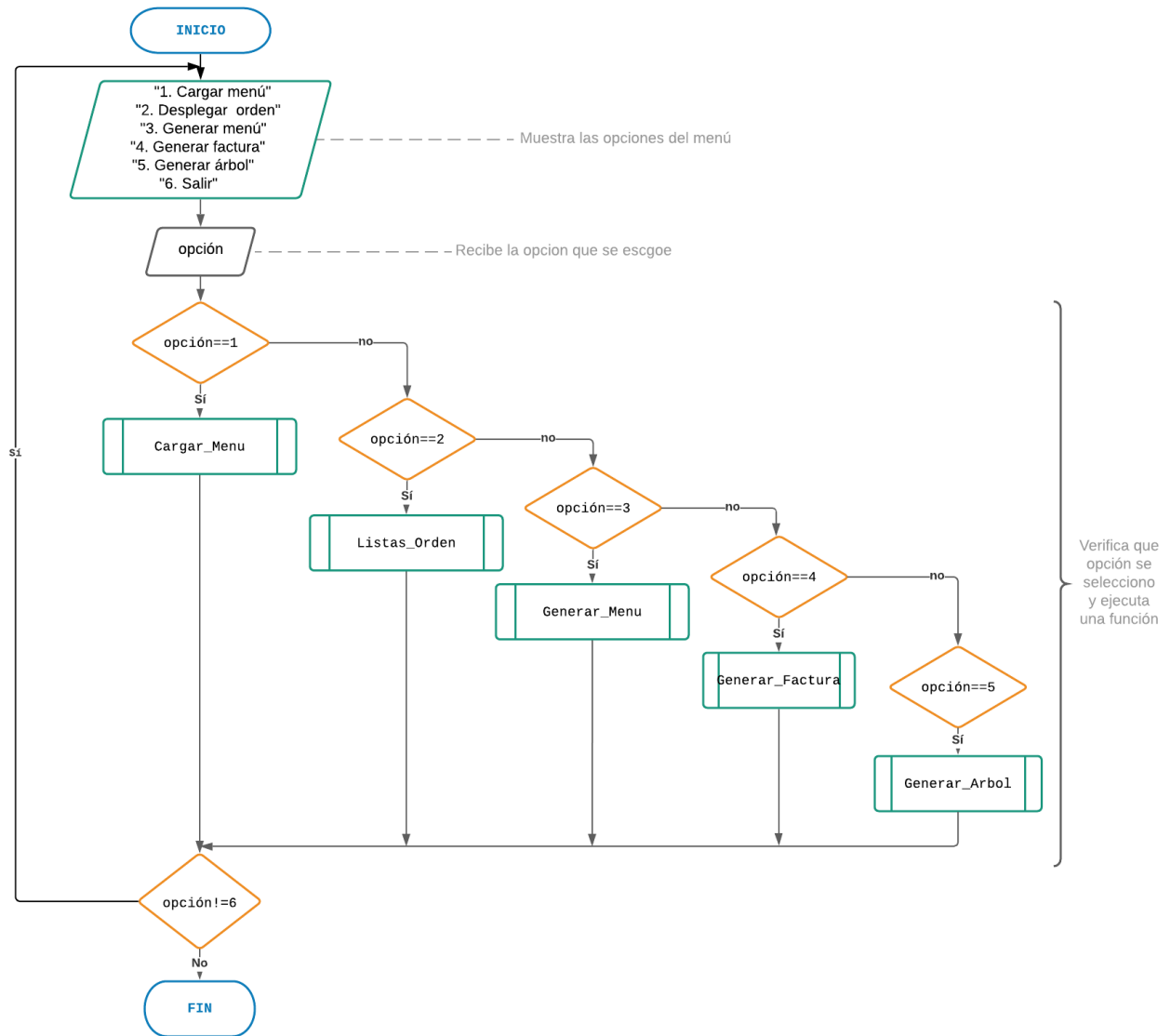
En cualquiera de los sistemas que vaya a ejecutar la aplicación se debe contar con un navegador que soporte HTML 5, para poder visualizar el reporte.

- Python en la versión 3.8.1 en adelante, puede descargarlo si no lo tiene desde su pagina oficial: <https://www.python.org/downloads/>.
- Graphviz puede descargarlo desde su pagina oficial: <https://www.graphviz.org/download/>.
- Paquete de python graphviz, si no se tiene este paquete puede instalarlo ejecutando el siguiente comando desde una terminal.

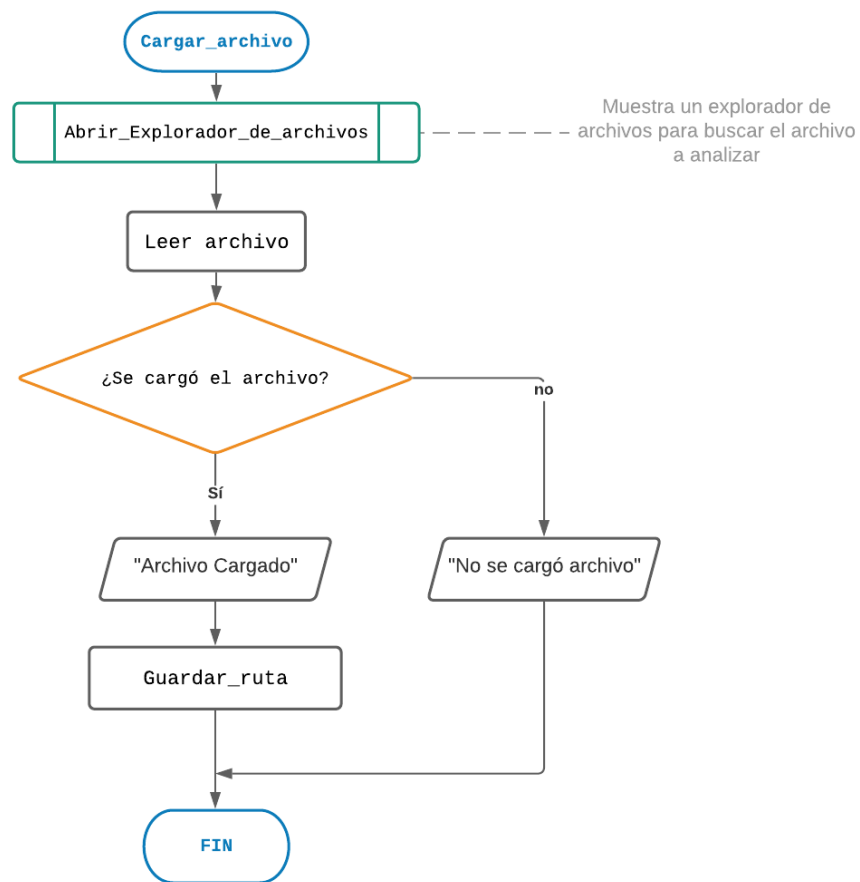
A screenshot of a terminal window with a dark background and light blue text. The command 'pip install graphviz' is entered. The window has three colored window control buttons (red, yellow, green) in the top left corner.

```
pip install graphviz
```

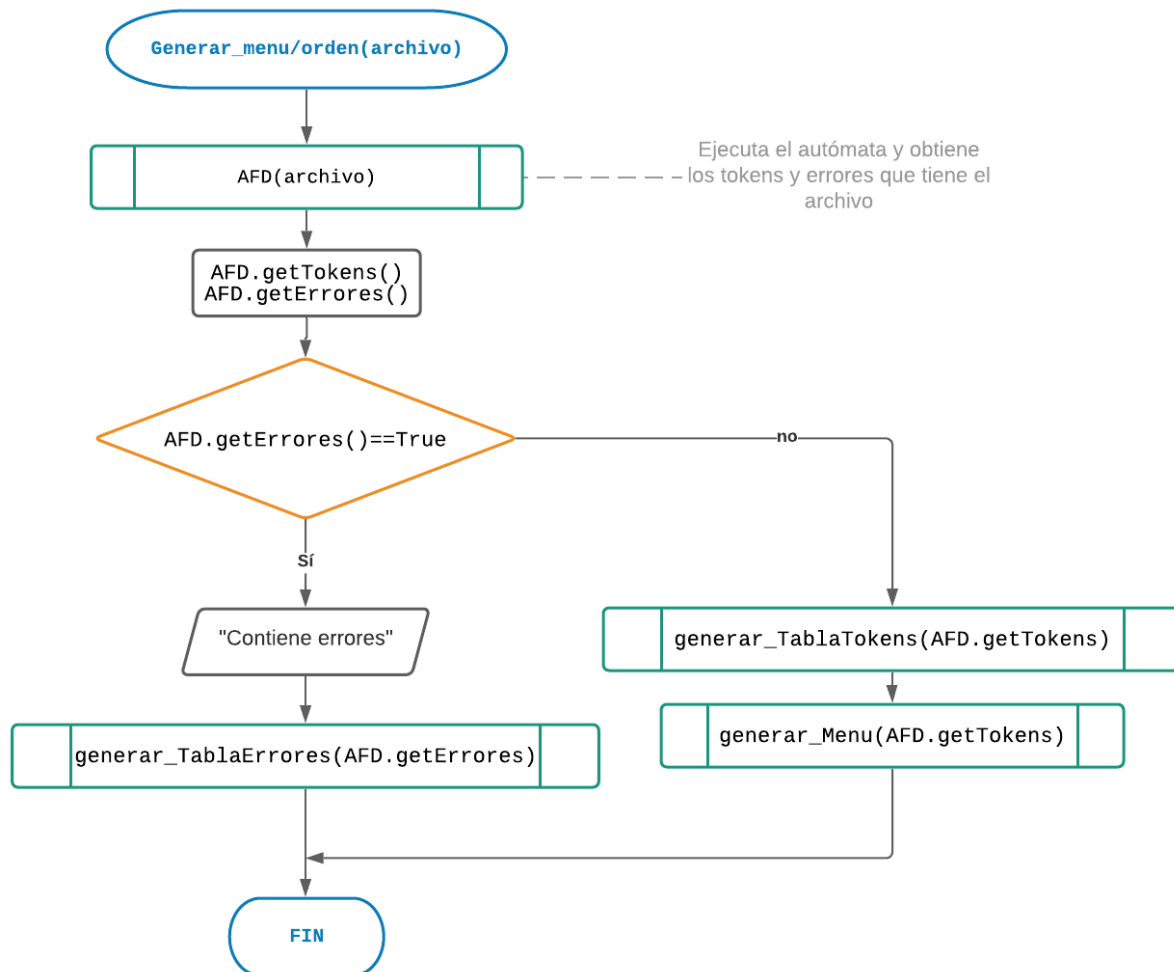
Menú



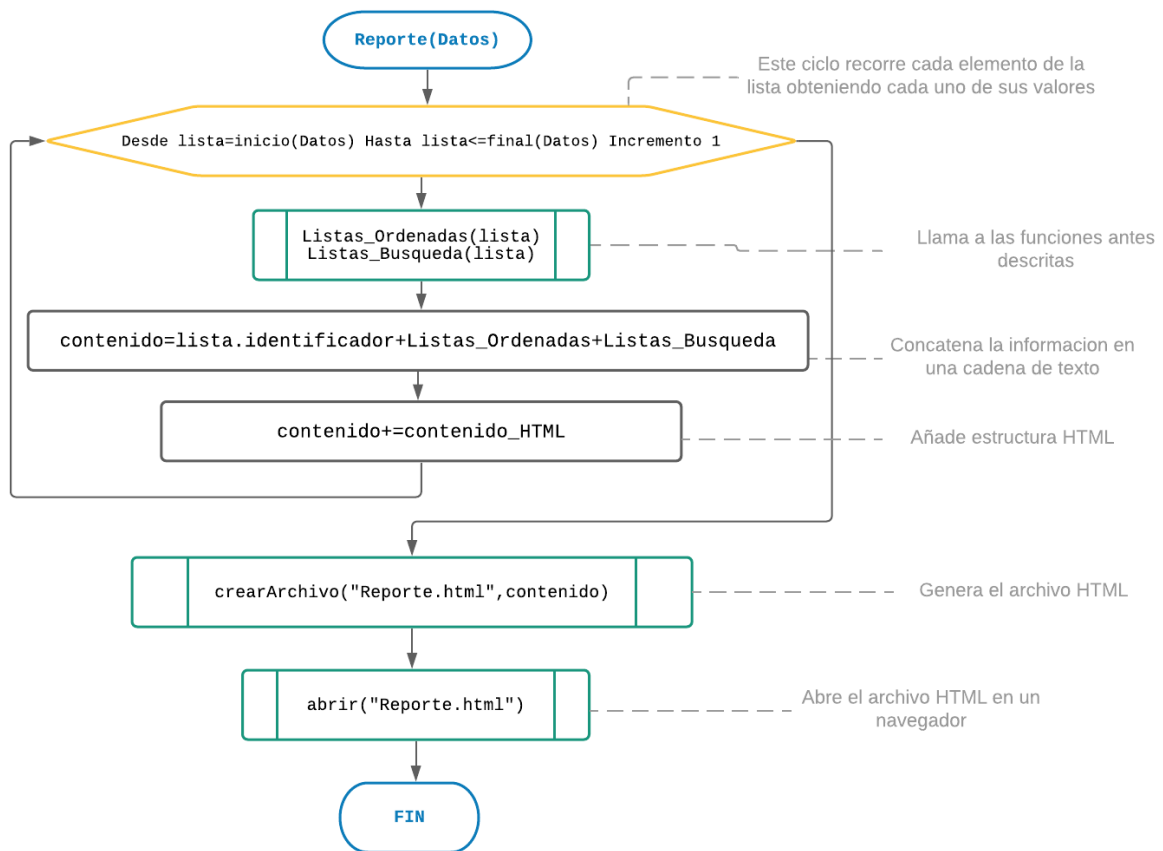
Cargar menú y orden:



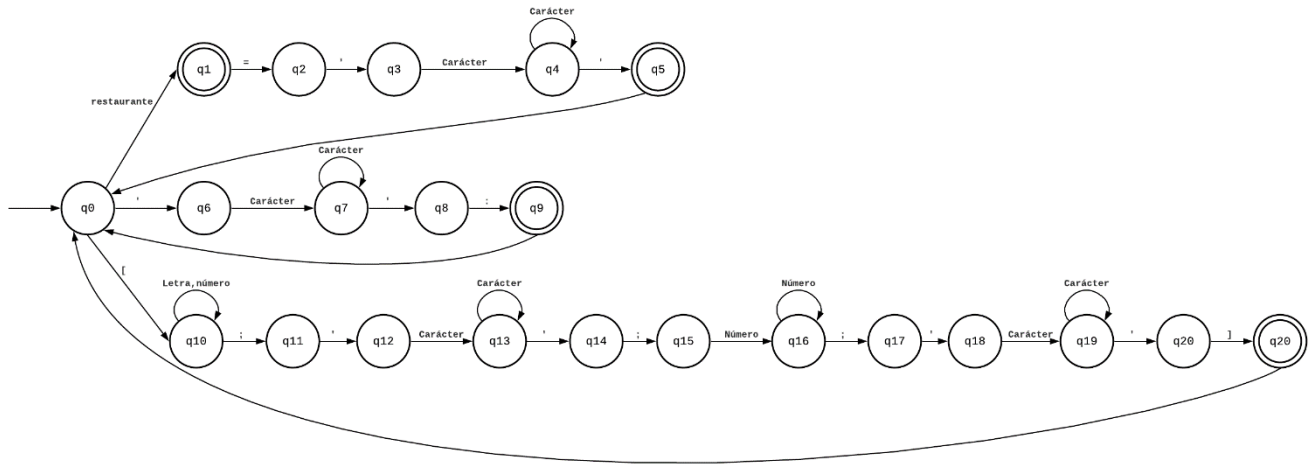
Generar menú y orden:



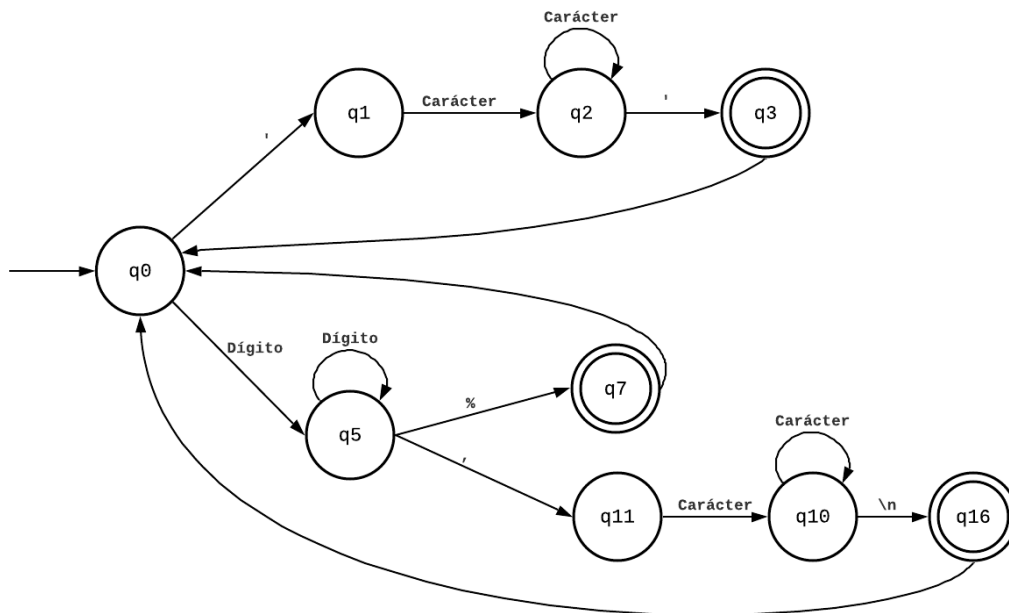
Generar tablas de tokens/errores:



AUTÓMATA FINITO DETERMINIS PARA LEER LA ESTRUCTURA DEL MENÚ



AUTÓMATA FINITO DETERMINISTA PARA LEER LA ESTRUCTURA DE LA ORDEN



PARADIGMAS DE PROGRAMACIÓN UTILIZADAS

Programación estructurada

```
def analizar(self):
    estado=0
    posicion=0
    columna=1
    string=""
    noIdentificados="~!@#$%^&*( )\, _+-|/¿i?{[]}'."
    longitud=len(self.texto)

    while posicion<longitud:

        caracter=self.texto[posicion]
        if estado==0:
            if caracter=="r":
                estado=1
                string+=caracter
                posicion+=1
                columna+=1

            elif caracter==" ":
                posicion+=1
                columna+=1

            elif caracter=="\n":
                posicion+=1
                self.Linea+=1
                columna=1

            elif caracter=="'":
                estado=5
                posicion+=1
                columna+=1
                self.contadorSecciones+=1

            elif caracter=="[" :
                estado=8
                posicion+=1
                columna+=1

            else:
                posicion+=1
```

Programación procedimental

```
def buscarReservada(self):
    encontrado=False
    for buscar in self.ListaTokens:
        if buscar.lexema=="restaurante":
            encontrado=True

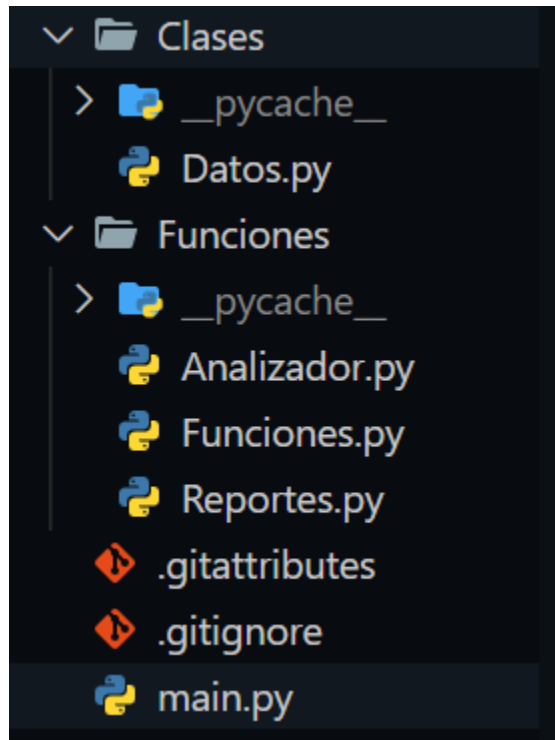
    if encontrado==False:
        aux=error("restaurante", "0", "0", "No existe la palabra reservada")
        self.ListaErrores.append(aux)

def buscarIdentificador(self,id):
    encontrado=False
    for buscar in self.ListaTokens:
        if buscar.lexema==id:
            encontrado=True

    return encontrado

def imprimirTokens(self):
    cont=1
    for tokens in self.ListaTokens:
        print(cont,tokens)
        cont+=1
```

Programación modular



```
from Funciones import Analizador
from Funciones import Funciones
from Funciones import Reportes
```

Programación Orientada a Objetos

```
class datos:
    def __init__(self, lexema, linea, columna, token):
        self.lexema = lexema
        self.linea = linea
        self.columna = columna
        self.token = token

    def __str__(self):
        string = str("Lexema: ") + str(self.lexema) + str(" Linea: ") + str(self.linea) + str(" Columna : ") + str(self.columna) + str(" Token: ") + str(self.token)
        return string

class error:
    def __init__(self, lexema, Linea, columna, descripcion):
        self.lexema = lexema
        self.Linea = Linea
        self.columna = columna
        self.descripcion = descripcion

    def __str__(self):
        string = str("Linea: ") + str(self.Linea) + str(" Columna : ") + str(self.columna) + str(" Carácter: ") + str(self.lexema) + str(" Descripción: ") + str(self.descripcion)
        return string
```

Estructura y formato de los archivos

Los archivos deberán ser guardados con la extensión. lfp

Menú

Nombre del restaurante:

El nombre del restaurante vendrá entre comillas simples, después de la palabra restaurante y del signo igual.

```
restaurante='NOMBRE DEL RESTAURANTE'
```

Nombre de sección:

Vendrá el nombre de la sección entre comillas simples, seguido de dos puntos y en las líneas siguientes aparecerán las opciones que tiene dicha sección.

```
'NOMBRE DE SECCIÓN':
```

Opciones del menú:

Las opciones del menú que tiene cada sección (bebidas, platos principales, postres, etcétera) vendrán entre corchetes uno en cada línea. Donde vendrá primero su identificador único, nombre, seguido del precio en quetzales y por último la descripción. Esta información vendrá separada por puntos y comas. El nombre y la descripción vendrán entre comillas simples.

```
[ IDENTIFICADOR, 'NOMBRE';PRECIO;'DESCRIPCIÓN' ]
```

Ejemplo de un archivo:

```
restaurante='Restaurante LFP'

'Bebidas':
[bebida_1;'Bebida #1';11.0;'Descripción Bebida 1']
[bebida_2;'Bebida #2';10.50;'Descripción Bebida 2']

'Desayunos':
[d1;'Desayuno 1';45.00;'Descripción Desayuno 1']
[d2;'Desayuno 2';40.0;'Descripción Desayuno 2']
[d3;'Desayuno 3';35.0;'Descripción Desayuno3']

'Postres':
[pos_001;'Postre 1';25.0;'Descripción Postre 1']
```

Pedidos

En cada línea vendrá cada una de las opciones pedidas, primero aparecerá la cantidad pedida seguido del identificador de la opción separados por una coma.

Ejemplo

En el siguiente ejemplo se muestra la orden de 2 opciones del menú. Con una propina del 8%. Tomar en cuenta que en la 1era línea, vienen los datos generales del cliente y el porcentaje de la propina; Nombre del cliente, NIT, dirección y Propina.

```
'XX', 'xx', 'xx', 8%
2,bebida_1
4, pos_001
```

CONCLUSIÓN:

- Es necesario establecer un formato y estructura al archivo que se desea analizar y procesar ya que con ello se pueden crear algoritmos que ayudaran a realizar tales acciones.
- El explorador de archivos hace más fácil la búsqueda y carga de archivos.
- Los reportes permiten visualizar de forma detallada cierta información.