

Universidad de San Carlos de Guatemala  
Facultad de ingeniería  
Ingeniería en Ciencias y Sistemas  
Lenguajes Formales y de Programación



## MANUAL TECNICO

Leonel Antonio González García 201709088

Guatemala 03 de marzo de 2024

## MANUAL TECNICO

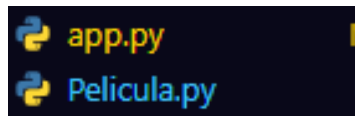
Con el uso de los paradigmas de programación orientado a objetos y la programación imperativa se desarrolló el programa solicitado por la empresa para el manejo de la información de películas que son leídos al cargar un archivo, haciendo uso de las clases y los métodos para que el programa cumpla con su funcionamiento de manera eficiente.

Siempre al inicio del programa encontraremos las librerías que nos ayudaran con la creación de las funciones que se necesitan.

La librería tkinter nos sirve para el uso del explorador de archivos.

```
from tkinter import filedialog
from tkinter import Tk
from Pelicula import *
from graphviz import Digraph
```

Para el desarrollo óptimo de este programa se recurrió al uso de clases.



### Clase app.py

En esta clase están los métodos que ayudan al funcionamiento óptimo del programa como la carga del archivo, es nuestra clase principal en la que se corre el programa.

```
def CargarArchivo():

    try:
        root = Tk()
        root.withdraw()
        root.attributes("-topmost", True)
        #Abre Ventana para Buscar el archivo .lfp
        archivo = filedialog.askopenfilename()
        print("Archivo Seleccionado", archivo)

        #Abre el archivo
        archivo_texto = open(archivo, 'r', encoding="utf8")
        #print(archivo_texto)
        #print("muestral")

        #Contenido del archivo leído
        texto = archivo_texto.read()
        #print(texto)
        global datos
        datos = texto
        root.destroy()
        #diccionario()
        #mostrarPelículas()
    except FileNotFoundError:
        print("El archivo no se cargó correctamente, vuelva a intentarlo.")
```

### Método diccionario()

En este método se crea un diccionario que contiene la información del archivo cargado, en este se agregan las películas que vienen en el archivo y se guardan para poder manejar toda la información.

```
def diccionario():
    info = datos.split("\n")
    for s in info:
        diccionario = s.split(";")

        peli = {
            "nombre:" +diccionario[0],
            "reparto:" +diccionario[1],
            "año:" +diccionario[2],
            "genero:" +diccionario[3]
        }
        lista_peliculas.append(peli)
    print(lista_peliculas)
```

### Método mostrarPeliculas()

Los métodos mostrar sirven para que el usuario pueda obtener una vista de la información de las películas y los actores más ordenada.

```
def mostrarPeliculas():
    #print(lista_peliculas)
    info = datos.split("\n")
    # con un for itero la cadena de texto y hago que guarde en una pos
    for i in info:
        diccionario = i.split(";")
        print("-----Película-----")
        print("Nombre:" +diccionario[0])
        print("Reparto:" +diccionario[1])
        print("Año:" +diccionario[2])
        print("Género:" +diccionario[3])
```

### Método filtrarActor()

Los métodos filtrar ayudan a la búsqueda de información de una forma más rápida por medio de entradas de búsqueda proporcionadas por el usuario se muestra la información de una forma más ordenada y eficiente.

```
def filtrarActor():
    entrada_actor = input("Ingrese el Nombre del Actor:... ")
    actor_de_pelicula = []

    for pelicula in datos.split("\n"):
        datos_pelicula = pelicula.split(';')
        actores = datos_pelicula[1].split(',')

        if entrada_actor in actores:
            actor_de_pelicula.append(datos_pelicula[0])

    if actor_de_pelicula:
        print("Las películas en las que está " + entrada_actor + " son:...")
        for pelicula in actor_de_pelicula:
            print(pelicula)
    else:
        print("No se encontraron películas en las que esté " + entrada_actor)
```

## Método grafica()

En este método se crea el diseño de la gráfica que contiene la información del archivocargado y la muestra de manera ordenada.

```
def grafica():
    info = datos.split("\n")

    lista_actores = {}
    for pelicula_info in info:
        # Dividir la información de la película por ';'
        datos_pelicula = pelicula_info.split(';')
        nombre_pelicula = datos_pelicula[0]
        actores = datos_pelicula[1].split(',')
        año = datos_pelicula[2]
        genero = datos_pelicula[3]
        for actor in actores:
            actor = actor.strip()
            if actor not in lista_actores:
                lista_actores[actor] = []
            lista_actores[actor].append((nombre_pelicula, año, genero))

    graph = Digraph()

    for actor, peliculas in lista_actores.items():
        graph.node(actor, shape="circle", style = "filled", fillcolor="blue", color = "skyblue")
        for pelicula in peliculas:
            nombre_pelicula = f"{pelicula[0]} ({pelicula[1]}) - {pelicula[2]}"
            graph.node(nombre_pelicula, shape="circle", style = "filled", fillcolor="orange", color = "skyblue")

    for actor, peliculas in lista_actores.items():
        for pelicula in peliculas:
            nombre_pelicula = f"{pelicula[0]} ({pelicula[1]}) - {pelicula[2]}"

            graph.edge(actor, nombre_pelicula)

    graph.attr(rankdir='LR')
    graph.render("Gráfica", format="pdf", cleanup=True)

    print("Gráfica creada como 'Gráfica.pdf'")
```

### Clase Película.

Esta clase sirve para definir la película y sus atributos en el método constructor.

```
class Pelicula:  
    def __init__(self, nombre, actor, año, genero):  
        self.nombre = nombre  
        self.actor = actor  
        self.año = año  
        self.genero = genero
```

Fin del Programa