MANUAL TÉCNICO

DENNY ALEXANDER CHALI MIZA

201807154

24/02/2023

ÍNDICE DE CONTENIDOS

1. Objetivos

La creación del documento tiene como propósito el de mostrar como fue diseñado el sistema y como interactuar con el programa para su mejoramiento o actualización.

Objetivos específicos

- Guía de instalación del sistema
- Requisitos para la instalación del sistema
- Mostrar el código fuente para una posible actualización del sistema.

2. Alcance

Este documento esta dirigido a: programadores o personas con conocimientos básicos en programación.

3. Requerimientos del sistema:

Para el correcto funcionamiento del programa se recomienda tener instalado en el equipo Python 3.8 o superior, la cual necesita como mínimo los siguientes requisitos.

- Procesador: Intel Pentium III o equivalente a 800 MHz
- Memoria RAM: 1 GB
- Espacio de disco: 1GB de espacio libre

4. Herramientas utilizadas para el desarrollo.

Editor de texto: Visual Studio Code v1.70

5. Descripción de la aplicación

Se desarrolló una aplicación que permite la lectura de un archivo de texto plano que contiene diferentes listas de valores separados por punto y coma. Estos valores contienen información como nombre de una película, actores que participan en ella, año de estreno y género de esta. El programa analiza los datos de los datos de cada película para representar de una forma más simple la información relacionada con los datos brindados.

Este software se desarrolló en el lenguaje Python, contando con múltiples opciones para la visualización e interacción con los datos almacenados en el sistema.

6. Desarrollo de la aplicación:

Los archivos en la carpeta contenedora son las que se encargan de la ejecución de cada una de las funciones del programa.

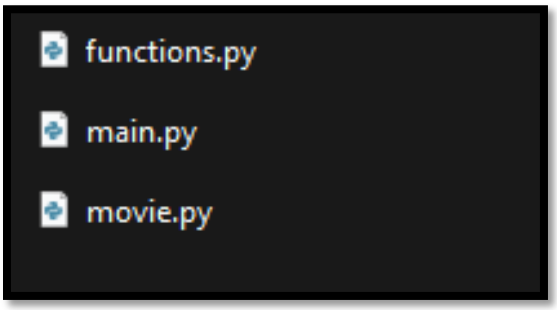

Figura 1. Archivos del sistema.
Fuente: Elaboración propia.

7.1 main

Descripción:

Se encarga de iniciar el programa, mostrar los mensajes y menús iniciales del sistema. También se encarga de crear la lista global que contendrá cada uno de los datos de las películas.

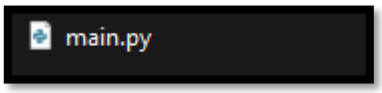

Figura 2. Archivos del sistema.
Fuente: Elaboración propia.

Código:

```
import msvcrt
from functions import *
movies = []

def run():
   print("""
Lenguajes formales y de programación
Sección B+
Carné: 201807154
Desarrollador: Denny Alexander Chalí Miza\n
Presiona cualquier tecla para continuar...\n""")
```

```
    #Esperar hasta que se presione una tecla
    msvcrt.getch()
    end = False
    selection = 0

    while not end:
        print("\n---------- Menú ----------\n  1. Cargar archivo\n  2. Gestionar películas\n  3. Filtrar\n  4.
 Gráfica\n  5. salir")
        selection = pedirNumeroEntero()

        if selection == 1:
            contentFile = leerArchivo()
            if contentFile != None:
                analizador(contentFile, movies)
            else:
                print("Datos no cargados")

        elif selection == 2:
            gestionar(movies)

        elif selection == 3:
            filtrar(movies)

        elif selection == 4:
            graficar(movies)

        elif selection == 5:
            print("Finalizando programa...")
            end = True
        else:
            print("Error, ingrese una opción correcta.")

 if __name__ == '__main__':
    run()
```

7.2 movie.

Descripción:

Para la fácil manipulación de los datos, se implementó la creación de la clase Movie teniendo como atributos de inicio el nombre, año y género. Para la creación de esta clase se hizo uso de la programación Orientada a Objetos.

Para su edición y visualización de datos se crearon los métodos set y get respectivamente.

Figura 3. Archivo movie.
Fuente: Elaboración propia.

Código:

```
class Movie:

    def __init__(self, nombre, año, genero):
        self.nombre = nombre
        self.año = año
        self.genero = genero
        self.actores = []

    def getNombre(self):
        return self.nombre

    def getAnio(self):
        return self.año

    def getGenero(self):
        return self.genero

    def getActores(self):
        return self.actores

    def setNombre(self, nombre):
        self.nombre = nombre

    def setAnio(self, anio):
        self.año = anio

    def setGenero(self, genero):
        self.genero = genero

    def setActores(self, actores):
        self.actores = actores

    def addActor(self, actor):
```

```
            self.actores.append(actor)

        def actorsClearList(self):
            self.actores.clear()
```

7.3 Functions.

Descripción:

Contiene y ejecuta cada una de las funciones de los submenús, los cuales son: Gestionar películas, Filtrar, Graficar y finalizar el programa. En esta sección del programa se hizo uso de la programación procedural debido a que fue más fácil de resolver el problema dividiendo cada funsión.

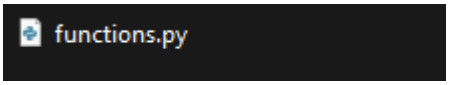

Figura 4. Archivos functions.
Fuente: Elaboración propia.

Código:

```
from tkinter import filedialog
import graphviz
from graphviz import nohtml
from movie import Movie

def pedirNumeroEntero():
    correcto=False
    num=0
    while(not correcto):
        try:
            num = int(input("Introduce una opción: "))
            correcto=True
        except ValueError:
            print('¡Error, introduce un numero entero!')
    return num

def leerArchivo():
    #obtenemos la direccion local del archivo
    root = filedialog.askopenfilename(title= "Abrir Archivo", filetypes=(("lfp","*.lfp"),("Todos los
archivos","*.*")))
    if root != "":
        try:
```

```
            #encoding utf-8
            file = open(root,'r',encoding='utf-8')
            contentFile = file.read()
            file.close()

            return contentFile

        except:
            print("Ha ocurrido un error")
            file.close()

    return None

def analizador(contentFile, movies:list):
    errores_detectados = False
    #quitamos los espacios al incio y al final de la cadena
    data = contentFile.strip()
    #creamos una lista con cada fila
    lines = data.split("\n")
    n = 0
    #recorremos la lista de lineas
    for line in lines:
        n += 1
        aux = line.split(";")
        if len(aux) == 4:
            try:
                nameMovie = aux[0].strip()
                movie = verificarRepetido(nameMovie, movies)
                if not(movie):
                    movie = Movie(nameMovie, aux[2].strip(), aux[3].strip())
                    actores = aux[1].split(",")
                    for actor in actores:
                        movie.addActor(actor.strip())
                    movies.append(movie)
                else:
                    movie.setAnio(aux[2].strip())
                    movie.setGenero(aux[3].strip())
                    actores = aux[1].split(",")
                    movie.actorsClearList()
                    for actor in actores:
                        movie.addActor(actor.strip())
            except:
                errores_detectados = True
                print("¡Ha ocurrido un error")
        else:
            errores_detectados = True
            print("Datos erroneos en la línea: " + str(n))
```

```
    if errores_detectados:
        print("Archivo Cargado con errores detectados")
    else:
        print("Archivo Cargado correctamente")

def verificarRepetido(pelicula, movies:list):
    for movie in movies:
        if movie.getNombre() == pelicula:
            return movie

    return None

def gestionar(movies:list):
    end = False
    selection = 0
    while not end:
        print("\n--------- Gesionar películas ---------\n 1. Mostrar películas\n 2. Mostrar actores\n 3.
Regresar")
        selection = pedirNumeroEntero()
        if selection ==1:
            mostrarPeliculas(movies)
        elif selection == 2:
            mostrarActores(movies)
        elif selection == 3:
            end = True
        else:
            print("Error, ingrese una opción correcta.")

def mostrarPeliculas(movies:list):
    i = 1
    print("\n--------- Películas Cargadas ---------")
    for movie in movies:
        movie:Movie
        print(str(i) + ". " + movie.getNombre()+ " | Año: " + movie.getAnio() +" | Género: " +
movie.getGenero()+"\n")
        i += 1

def mostrarActores(movies:list):
    print("\n-------- Seleccione película --------")
    i = 0
    for movie in movies:
        i += 1
        print(str(i) + ". " + movie.getNombre())

    num = pedirNumeroEntero()
    if num <= i and num > 0:
```

```
        movie:Movie = movies[num -1]
        actors = movie.getActores()
        i += 1
        print("\n"+movie.getNombre()+ " | Año: " + movie.getAnio() +" | Género: " +
movie.getGenero())
        print("Actores: ")
        j = 1
        for actor in actors:
            print("\t" + str(j) +". "+actor)
            j += 1
    else:
        print("¡Ingrese una opción correcta!")


def filtrar(movies):
    end = False
    selection = 0

    while not end:
        print("\n---------- Filtrar por ----------\n  1. Actor\n  2. Año\n  3. Género\n  4. Regresar")
        selection = pedirNumeroEntero()

        if selection == 1:
            filtrarActor(movies)

        elif selection == 2:
            filtrarAnio(movies)

        elif selection == 3:
            filtrarGenero(movies)

        elif selection == 4:
            end = True
        else:
            print("Error, ingrese una opción correcta.")

def filtrarActor(movies:list):
    print("\n-------- Seleccione actor --------")
    i = 0
    actores = []
    for movie in movies:
        for actor in movie.getActores():
            if not(actor in actores):
                actores.append(actor)
                i += 1
                print(str(i) + ". " + actor)
```

```
    num = pedirNumeroEntero()
    if num <= i and num > 0:
        seleccionado = actores[num-1]
        print("\nActor: " + seleccionado)
        print("Peliculas: ")
        j = 1
        for movie in movies:
            if seleccionado in movie.getActores():
                print("\t"+ str(j)+". " + movie.getNombre())
                j += 1
    else:
        print("¡Ingrese una opción correcta!")

def filtrarAnio(movies):
    print("\n-------- Seleccione año --------")
    i = 0
    anios = []
    for movie in movies:
        if not(movie.getAnio() in anios):
            anios.append(movie.getAnio())
            i += 1
            print(str(i) + ". " + movie.getAnio())

    num = pedirNumeroEntero()
    if num <= i and num > 0:
        seleccionado = anios[num -1]
        print("\nPeliculas del año: " + seleccionado)
        j = 1
        for movie in movies:
            if movie.getAnio()== seleccionado:
                print(str(j)+". "+ movie.getNombre())
                j+=1
    else:
        print("¡Ingrese una opción correcta!")

def filtrarGenero(movies):
    print("\n-------- Seleccione Género --------")
    i = 0
    generos = []
    for movie in movies:
        if not(movie.getGenero() in generos):
            generos.append(movie.getGenero())
            i += 1
            print(str(i) + ". " + movie.getGenero())

    num = pedirNumeroEntero()
    if num <= i and num > 0:
```

```
            seleccionado = generos[num -1]
            print("\nPeliculas genero: " + seleccionado)
            j = 1
            for movie in movies:
                if movie.getGenero() == seleccionado:
                    print(str(j)+ ". "+ movie.getNombre())
                    j+= 1
        else:
            print("¡Ingrese una opción correcta!")

def graficar(movies:list):
    g = graphviz.Digraph('g', filename='Movies',
                         node_attr={'shape': 'record', 'height': '.1'})
    g.attr(rankdir='LR')

    actores = []
    for movie in movies:
        for actor in movie.getActores():
            if not(actor in actores):
                actores.append(actor)

    with g.subgraph(name='cluster_0') as c:
        c.attr(style='filled', color='white', margin= "40")
        for movie in movies:
            movie:Movie
            c.node(movie.getNombre(), nohtml(r'{ {<f0> '+ movie.getNombre() +' | {'+
movie.getGenero() +' | '+ movie.getAnio() +' } }}'), color = "blue")

    with g.subgraph(name='cluster_1') as c:
        c.attr(style='filled', color='white', margin = "40")
        for actor in actores:

            c.node(actor, nohtml('<f0>'+ actor))

    for movie in movies:
        movie:Movie
        movie_actors = movie.getActores()
        for actor in movie_actors:
            g.edge(movie.getNombre(), actor)
        #g.edge('peli1:f0', 'actor2:f0')

    g.view()
```