OBSERVACIONES DEL LA PRÁCTICA

Estudiante 1: Nicholas Barake Cod: 202020664

Estudiante 2: Jessed Domínguez Cod: 202011992

1. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **view.py** con el usuario?

El mecanismo de interacción con el usuario recae en el menú principal. Es la manera en la cual el usuario le solicita al programa una información, para que el código de view.py lo pueda escalar al controlador, al modelo y asímismo de regreso con la información solicitada. El menú le ofrece al usuario: cargar los datos (que es necesario para poder realizar las otras consultas), ver los top libros, ver los libros de un autor, ver libros por género, y salir del programa. En estas, el usuario debe ingresar una información (input) con la que el código debe trabajar para retornar la información o realizar la acción correspondiente (output).

Text

Description automatically generated

Deja de ejecutar el programa.

“5” para salir del programa. (Parar de ejecutar.)

“4” para consultar cuántos libros hay por género.  
“x” donde x es el parámetro del género(label) para la función del controlador countBooksByTag().

“2” para consultar los top libros.  
“x” donde x es el parámetro de los mejores x libros por promedio para la función del controlador getBestBooks().

Muestra cuantos libros hay del género x.

“3” para consultar los libros de autor particular.  
“x” donde x es el parámetro del nombre del autor para la función del controlador getBooksByAuthor ().

Muestra los libros del autor x.

Muestra los mejores x libros por promedio.

Muestra número de datos que se cargaron para libros, autores, géneros y tags género-libro.

“1” para solicitar carga de datos.

Output del programa

Input del usuario

1. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?

El archivo model.py almacena los datos dentro de un “catalogo” que es como un diccionario con las llaves de: books, authors, tags, y book\_tags. Cada llave tiene como valor correspondiente una lista vacía que luego será llenada cuando se carguen los datos. Esta lista es un tipo abstracto de datos de Lista, especificamente de tipo ARRAY\_LIST. La función que crea las listas está en la librería DISClib, en el archivo list.py.

1. ¿Cuáles son las funciones que comunican el **view.py** y el **model.py**?

Todas las funciones (excepto la de ordenar) en controller.py sirven para comunicar solicitudes en view.py a model.py. Las funciones de controller.py importan las funciones de model.py para hacer la consulta de los datos en model.py con los parámetros dados por el usuario en view.py. La mayoría de las funciones en view.py van a importar funciones de controller.py, controller.py de model.py. No se considera necesario listar todas las funciones que comunican ambos archivos ya que son demasiados.

1. ¿Cómo se crea una lista?

Las listas se crean con la función lt.newList(). Donde, “lt” es el “as” del archivo list.py de la librería DISClib. Este módulo contiene la función que crea una lista vacía de tipo TAD lista, que luego será utilizado en model.py para crear el catálogo de los libros.

1. ¿Qué hace el parámetro **cmpfunction=None** en la función **newList()**?

Su función es comparar los valores de la nueva lista con la función que se le indica. En list.py cmpfunction es igual a None, porque aun no se está creando la nueva lista. Pero en model.py, sí entra como parámetro, por ejemplo, “cmpfucntion=compareauthors” donde “compareauthors” es otra función dentro de model.py que compara los nombres de los autores en la lista.

1. ¿Qué hace la función **addLast()**?

Añade un elemento en la última posición de la lista dada por parámetro. Por lo que aumenta el tamaño de la lista en 1.

1. ¿Qué hace la función **getElement()**?

Obtiene el elemento de la lista en la posición x dada por parámetro, recorriendo la lista hasta llegar a la posición solicitada. La posición debe ser mayor que cero, menor o igual que la longitud de la lista, y la lista no debe estar vacía.

1. ¿Qué hace la función **subList()**?

Retorna una sublista de la lista dada por parámetro, que contiene todos los elementos de la lista desde la posición x dada por parámetro hasta el número de elementos x, también dado por parámetro, que se quieran agregar a la sublista. Es decir, crea una mini-copia de la lista y la retorna.

1. ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**?

Se pudo observar que, al hacer el cambio del tipo de estructura de lista, al encontrar un elemento se tardó más en SINGLE\_LINKED que en ARRAY\_LIST. Eso se debe a las distancias entre elemento. Es decir, como los SINGLE\_LINKED no están pegados en memoria se demora más en recorrer que en los ARRAY\_LIST. Pero para encontrar por ejemplo el último elemento de la lista, el SINGLE\_LINKED es útil porque puede ir al último elemento sin recorrer.