OBSERVACIONES DEL LA PRACTICA

Nicolas Saavedra - 20212963

Ernesto Perez  - 202112530

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

Los mecanismos de input usados en el **view.py** corresponden a aquellos presentados en el menú principal con los números 1-5. Adicionalmente, algunas funciones del menú principal reciben como inputs números o nombres de autores o etiquetas que se quieran busca.

Los outputs que tiene el view.py ocurren al correr el programa mostrando el menú principal y seleccionar una opción diferente de 0 en el menú principal y dar la información adicional en caso de ser requerida. Se imprime la información solicitada por el usuario generalmente siendo una cadena seguida de información numérica.

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

Es un array list (Arreglo) que se conforma de un diccionario con las llaves “books”, “authors” ,”tags” y “book\_tags” donde el valor de cada llave es una lista con la información correspondiente a la llave.

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

Mediante las funciones del controller.py que son: “init Catalog”, “loadData”, “loadBooks”, “loadTags”,”loadBooksTags”, “sortBooks”, “getBooksByAuthor”, “getBestBooks” y “countBooksByTags”. Estas funciones son llamadas por view.py y llaman a model.py según sea necesario.

1. ¿Cómo se crea una lista?

La lista es creada mediante la funcion newList(), la cual llama internamente una funcion de la libreria para clasificarla si es una lista encadenada o un arreglo y devolver la que sea necesaria.

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

Coloca como valor por defecto de la funcion de comparacion de la lista, como nulo, lo cual es despues interpretado por la libreria como que utilize una funcion de comparacion de lista por defecto.

1. ¿Qué hace la funció **addLast()**?

La función recibe como parametro un elemento, y este es añadido al final de la lista dada. La implementacion es distinta dependiendo de el tipo de lista, ya que en un arreglo, utiliza la funcion append() de python para añadir un valor, mientras que en una lista encadenada, tiene que modificar el ultimo nodo para que apunte al nuevo ultimo nodo.

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

La funcion recibe como parametro la lista y una posicion en la que se desea buscar un elemento, luego recorre la lista y retorna el elemento siempre que la lista no este vacia.

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

La función recibe como parámetro un índice i en la lista y un número n de elementos, y esta retorna una lista con los elementos desde la posición i hasta la posición i + n. Ambas estructuras hacen el mismo proceso, llegando a la posición i e iterando hasta i + n, obteniendo cada elemento en cada iteración y añadiéndolo a una lista nueva. La diferencia es que mientras una lista normal puede simplemente conseguir la ubicación con el índice instantáneamente, una lista encadenada tiene que iterar a través de ella cada vez que necesita hallar un elemento.

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

El tiempo de carga de la informacion no se ve afectada por la estructura de datos sin embargo, el tiempo de busqueda de la informacion si sufre de un cambio notable en donde la lista encadenada es mas lenta que el arreglo.