Arquitectura Emmpresarial

Laboratorio 2

Intérpretes, canales de comunicación y memoria

Julián David Gutiérrez Vanegas

Agosto 2020

1. Introducción

En el siguiente artículo se busca presentar el modelo de una aplicación web, desarrollada con Java y gestionada y construida utilizando Maven y desplegada en Heroku. El objetivo de esta aplicación web es el cálculo de la media y la desviación estándar obtenidas de una serie de datos ingresados por el usuario en una página web.

Para la elaboración de este programa, fue necesario implementar una estructura de datos llamada "lista doblemente enlazada". Esta permitió almacenar y mantener los datos ingresados por el usuario en una estructura que otorga la ventaja de: agregar, eliminar, obtener y remplazar un dato en tiempo lineal. Lo cual, permite que el programa entregue una respuesta de manera más rápida al usuario.

Para la parte lógica del programa, se implementó una clase que es la encargada de: leer los datos, almacenarlos, realizar los cálculos correspondientes y mostrar o retornar la respuesta esperada.

Finalmente, para la comunicación con el usuario vía sitio web, se utilizó la librería Spark de Java, la cual nos permitió montar un servidor web ligero que se encargará de recibir los datos, enviárselos a la clase encargada de la lógica, para luego recibir la respuesta y mandarla al usuario.

A continuación, expondremos el modelo del programa dividiéndolo en sus componentes básicos: memoria, interprete y canales de comunicación. Después, se mostrarán unas pruebas de funcionamiento y, terminaremos, con un par de conclusiones.

2. Explicación del modelo

Dividiremos la explicación del modelo en tres partes: memoria, interprete y canales de comunicación.

2.1. Memoria

Para el almacenamiento de los datos ingresados por el usuario se utilizó una lista doblemente encadenada. Para la implementación de esta se comenzó con la creación de una interfaz que define qué funciones debe ofrecer una lista para el correcto desarrollo del programa. A continuación, se presenta el diseño de esta interfaz:

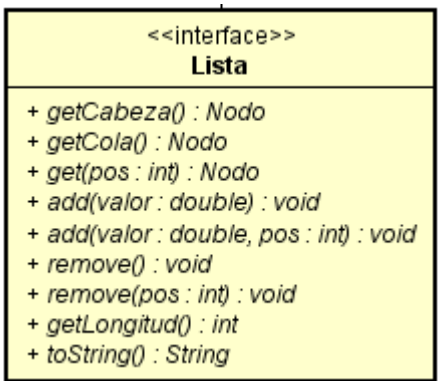

Figura 1: Interfaz de la lista

Una vez se tuvo implementada la interfaz lista que define lo que se necesita para el desarrollo del programa, se implementa la clase "LinkedList"que es la lista que se utilizó para el desarrollo del programa. Esta clase, además de implementar los métodos definidos en la interfaz, tendrá los atributos de: cabeza (objeto de tipo nodo que es primera posición de la lista), cola (último nodo de la lista) y longitud (número de elementos que están en la lista). La conexión entre todos los elementos de la lista se hace gracias a los atributos "siguientez .ªnterior"(ambos de tipo nodo) que se encuentran en todos los elementos nodo que componen la lista. Además de estos dos atributos, todos los nodos tienen un atributo "valor", el cuál es un "double", para que los cálculos tengan una mayor precisión. A continuación, se presenta completamente la estructura "listaïmplementada para este programa:

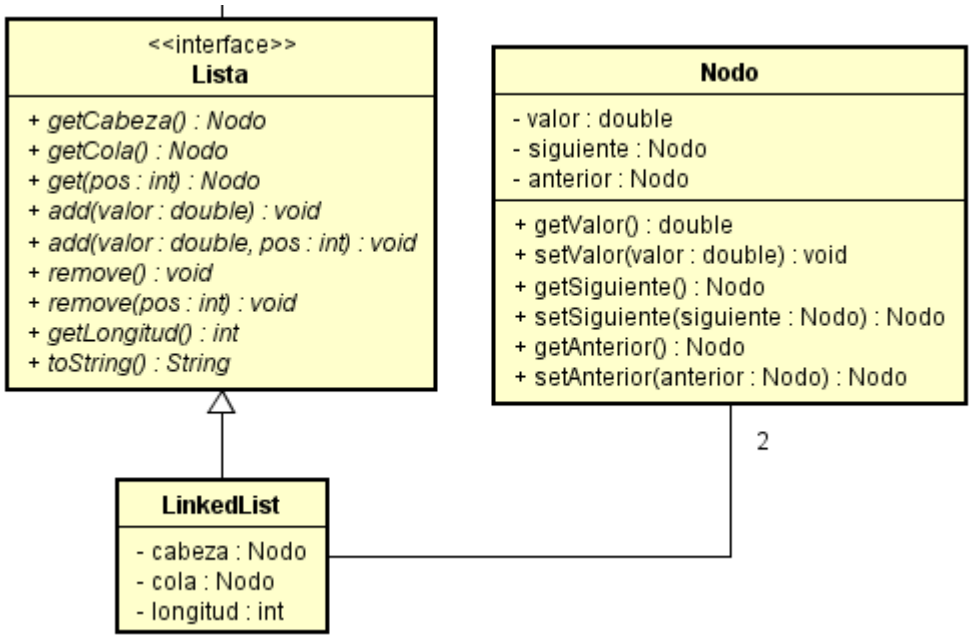

Figura 2: Implementacion de lista

Como podemos observar, este modelo, además de dar una mayor velocidad a la hora de administrar los datos que se ingresen, también es flexible. Esto permite que el programa, de ser necesario, utilice una implementación de "lista"distinta, que de tener cada uno de los métodos descritos en la interfaz bien implementados, no afectará el funcionamiento del programa.

2.2. Intérprete

Una vez se tuvo implementada la lista que se va a usar para almacenar los datos ingresados por el usuario, se implementó la clase que se encargará de la parte lógica de la aplicación. En esta clase se leen los datos que se encuentran en el archivo ingresado por usuario, se almacenan y, finalmente, se ejecutan todos los cálculos necesarios para entregarle en pantalla al usuario la media y la desviación estándar obtenidas. Este programa está dividido en dos métodos: el principal "main", el cual recibe la ruta del archivo donde se encuentran los datos que se usarán para realizar los cálculos y luego muestra la información del resultado en pantalla, y el método çalcularMediaYDesviacion"donde se realizan todas las operaciones. Este segundo método tiene dos diferentes implementaciones: una recibe los datos desde un archivo de texto plano, mientras que la otra recibe los datos ya almacenados previamente en una lista. A continuación, se presenta el diagrama de clases de todo el programa, donde se puede observar cómo interactúa esta clase principal con las demás:

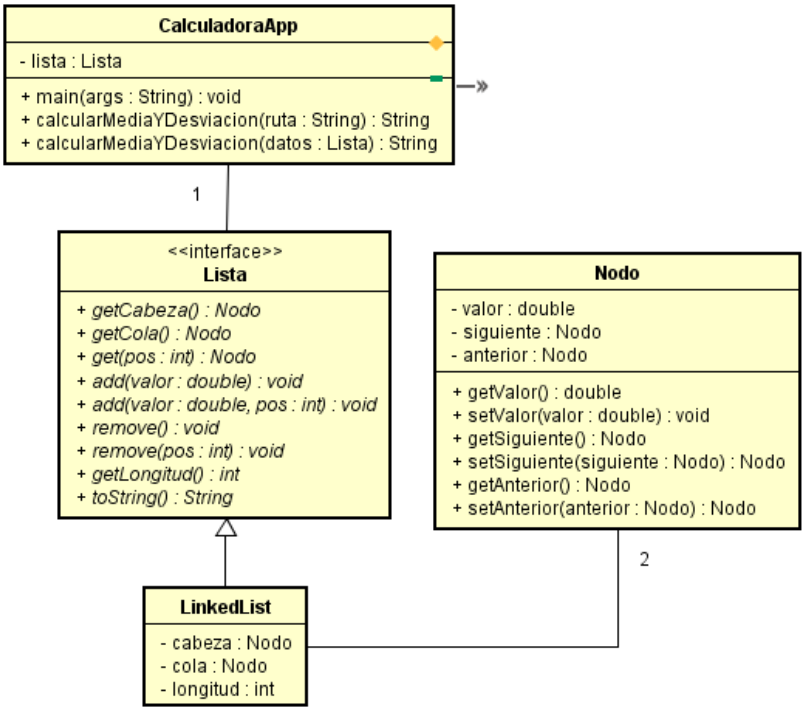

Figura 3: Diagrama de clases

2.3. Canales de comunicación

Por último, en la implementación del servicio web, se creó una clase, la cual sirve como canal de comunicación entre el usuario y el servidor, cuyo objetivo es: recibir solicitudes de tipo Post, que contengan los datos proporcionados por el usuario, y enviar las respuestas correspondientes a esas solicitudes. Para la implementación de esta clase se utilizó la librería Spark que nos permite levantar un servicio web ligero mediante: la elección de un puerto donde ejecutar el servicio, la selección de una carpeta de recursos y la implementación de acciones a realizar ante distintos tipos de solicitudes (get, post, delete, put, etc...). Al ser esta una aplicación muy pequeña, solo fue necesario implementar un método que se encargue de las solicitudes post. Este pequeño método lo único que hace es leer y almacenar los datos que llegan desde el usuario, enviarlos al método çalcularMediaYDesviacionz, cuando haya respuesta, mandarla al usuario. Cabe resaltar que también, en la parte del cliente, se implementó un pequeño método, utilizando Javascript, que es el encargado de enviar las solicitudes del usuario, para luego, cuando haya una respuesta del servidor, mostrar en pantalla la solución de la operación.

3. Pruebas

Para verificar que el programa esté funcionando de manera correcta se realizaron dos pruebas: una con números enteros y otra con números decimales. A continuación, se listan los datos utilizados:

Prueba 1	Prueba 2
160	15.0
591	69.9
114	6.5
229	22.4
230	28.4
270	65.9
128	19.4
1657	198.7
624	38.8
1503	138.2

Cuadro 1: Valores de las pruebas

Con estos datos se calculó la media y desviación estándar utilizando este programa y se compararon estos resultados con los resultados que arrojó otro programa matemático:

Prueba	Valor Esperado		Valor Arrojado	
	Media	Des.Est.	Media	Des.Est.
Prueba 1	550.60	572.03	550.60	572.03
Prueba 2	60.32	62.26	60.32	62.26

Cuadro 2: Resultados

Como se puede observar, los resultados fueron los mismos, lo que significa que el programa está funcionando correctamente.

4. Conclusiones

- Para la implementación de una lista no es necesario tener una clase compleja. Esta implementación se puede dividir en dos clases, teniendo que la clase lista solo referencie a los nodos de los extremos de la lista, mientras que cado uno de los nodos referencia a sus vecinos. Esta implementación se vuelve más eficaz si los nodos referencian a dos vecinos (el anterior y el siguiente), en lugar de solo referenciar a uno.
- Spark es una librería de Java que nos permite de manera muy fácil y rápida desarrollar aplicaciones web. Es una ideal si se quieren implementar servicios web pequeños que no necesiten configuraciones muy sofisticadas.

- Una aplicación web puede ser dividida fácilmente en los componentes más básicos de la computación: memoria, intérpretes y canales de comunicación.

5. Bibliografía

- Germán Camaño, G. C. (s. f.). Estructuras de datos: listas enlazadas, pilas y colas. Recuperado 13 de Agosto de 2020, de http://www.calcifer.org/documentos/librognome/glib-lists-queues.html
- Molina, I. F. (2017, 26 enero). Spark Framework. Recuperado 19 de Agosto de 2020, de https://www.adictosaltrabajo.com/2017/01/26/spark-framework/