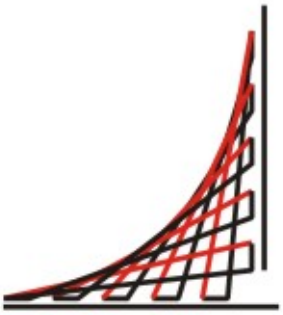
colorlinks, citecolor=black, filecolor=black, linkcolor=black, urlcolor=black



ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

AREP

Arquitecturas Empresariales

Segundo Laboratorio AREP

Author:
Juan Carlos García.

Agosto 2020

1. Introducción

A partir de la teoría básica sobre la teoría de listas encadenadas se realiza un programa que calcula el promedio y la desviación media de un conjunto de datos.

Este programa se configura como una aplicación web disponible por medio de la plataforma de **Heroku**, dando libertad a los usuarios de utilizar el servicio por un medio web.

En el siguiente documento se hace explicación de como fue diseñada la LinkedList encargada de conectar la información y el funcionamiento que se ve empleado en la conexión de los nodos para la comunicación del sistema a partir del lenguaje JAVA. También, se realiza una breve explicación de la teoría aplicada a los calculos de promedio y desviación media.

2. Descripción

2.1. Cómo se diseñó?

Principalmente partimos de la teoría conocida en las Linked Lists:

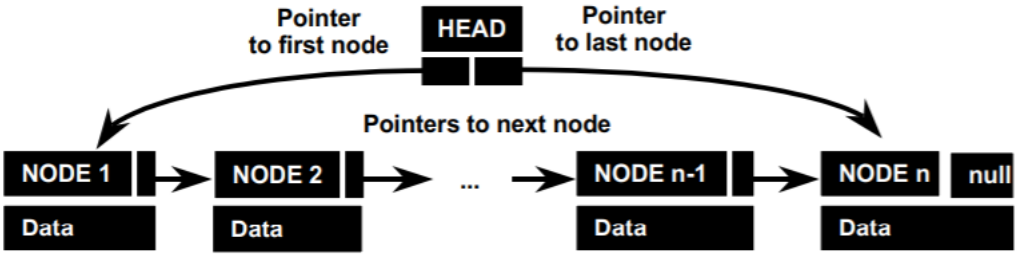


Figura 1: Linked List

En el caso de la programación, se planteó la existencia de punteros a los nodos **HEAD** y **TAIL**, con el fin de almacenar mayor información en la linked list y dar posibilidad a mejorar la extructura de la información.

Como podemos ver en la figura, los punteros únicamente realizan una conexión al nodo siguiente, ignorando por completo la existencia del anterior, es por este motivo que en mi diseño he realizado también un puntero a los nodos anteriores, permitiendo al nodo conocer cuales se encuentran en ambas direcciones a este.

Para Ejemplificar el como se ve la información para el código:

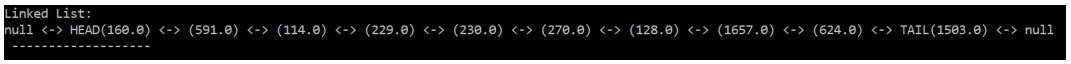


Figura 2: Linked List Example

Como se puede apreciar, se hace referencia también a los nodos izquierdos, permitiendo así una futura implementación a nuevos métodos de linked list.

Ahora, por parte del proceso de análisis a los calculos de media y desviación, tenemos la siguiente teoría:

MEAN:

$$x_{avg} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Figura 3: Media: Promedio

DEVIATION:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_{avg})^2}{n-1}}$$

Figura 4: Deviation

Ahora bien, para el caso que se pudo apreciar en en la Figura 2, los datos corresponden al caso de la columna 1 de la siguiente tabla:

Column 1	Column 2
Estimate Proxy Size	Development Hours
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

Table 1

Figura 5: Tabla de datos

Esta información tendría que dar el siguiente resultado:

Test	**Expected Value**		**Actual Value**	
	Mean	*Std. Dev*	*Mean*	*Std. Dev*
Table 1: Column 1	550.6	572.03		
Table 1: Column 2	60.32	62.26		

Figura 6: Datos Prueba: resultado

Información que se cumple a partir de la estructura generada para el calculo..

```
Desviacion
Sum: 5506.0
Mean: 550.6
Deviation: 572.026844746915

Desviacion Redondeada
Sum: 5506.0
Mean: 550.6
Deviation: 572.03
```

Figura 7: Desviación: Programa

Es posible hacer el valor más aproximado o con una cierta cantidad de cifras segun sea requerido, como es el caso, se puede apreciar que existen dos escrituras de la desviación. esto se hace a partir del método de redondeo expuesto más adelante.

```
public static String Round(double number, int nDigit) {
    double digits = Double.valueOf(Math.pow(10,nDigit));
    return String.valueOf((double)Math.round(number * digits) / digits);
}
```

La construcción para este funcionamiento se realizó a apartir de un código basado en partición de información, como ejemplo tenemos el siguiente fragmento de código correspondiente al calculo de la desviación:

```
public static String Deviation(MyOwnLinkedList linkedList){
    int size = linkedList.getSize();
    double mean = Double.valueOf(StatCalculator.Mean(linkedList));
    double Expo = Double.valueOf(StatCalculator.DeviationSum(linkedList,mean));
    double sqrt = Math.sqrt(Expo/(size-1));
    return String.valueOf(sqrt);
}
```

2.2. ¿Cómo se desplegó?

Principlamente se diseñó una estructura que permitiera generar la información HTML diréctamente en la clase, de este modo la respuesta sería sencilla.

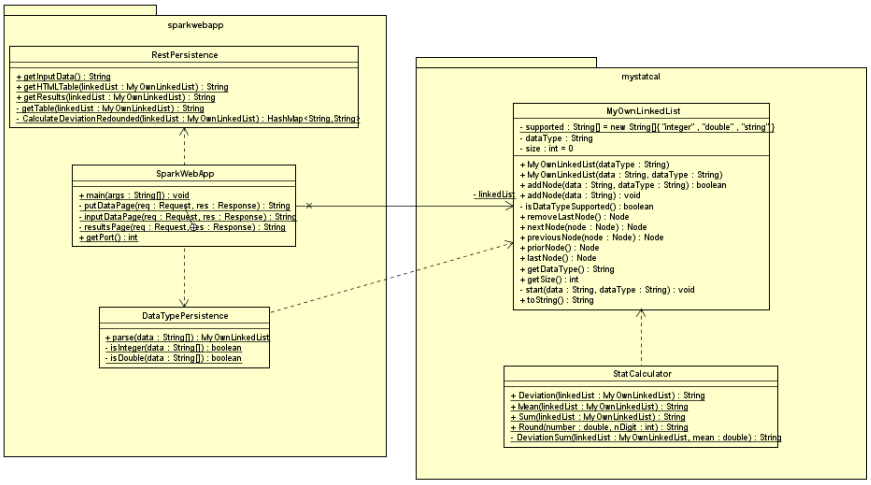


Figura 8: Diagrama simple de conexión

.

A partir de la siguiente Link se puede ver el despliegue de la aplicación por medio de Heroku. En este punto se puede notar que el diseño del Front puede ser mejorado, en este caso solo se presenta que los requerimientos funcionales se han alcanzado.

Para hacer el correspondiente cambio de la aplicación como un servicio web se hizo uso del **framework Spark**, de este modo se expandió la estructura del programa por medio del sistema de respuestas.

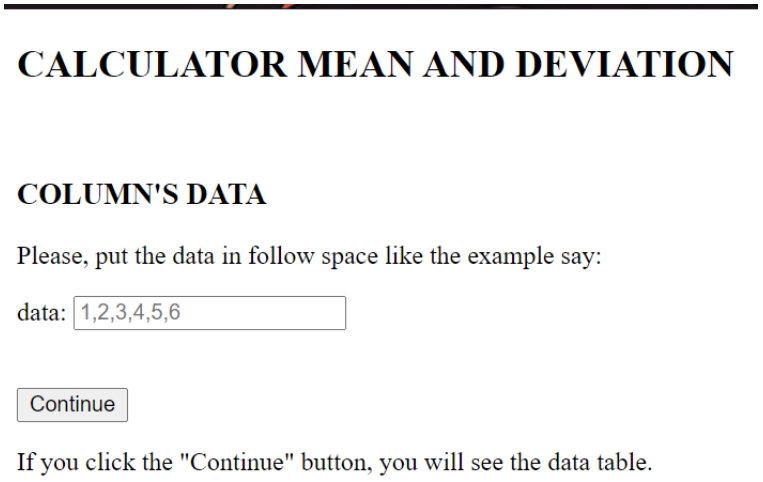


Figura 9: Primer boceto página despliegue

3. Pruebas

La construcción del código se realiza por medio de **Maven**, es por este motivo que las pruebas pueden ser compiladas directamente por este servicio.

esto se hace a partir del comando

```
mvn surefire:test
```

Como resultado poseemos la información de pruebas automatizadas realizadas en **Junit** que verifican la integridad del código y confirman un correcto funcionamiento en los casos de uso a la hora de realizar el uso del sistema.

4. Conclusiones

El sistema posee ciertas herramientas que permiten la implementación de una mejor estructura, del mismo modo, facilita la extensibilidad y es posible realizar nuevas funciones para la calculadora.

Su despliegue es funcional y puede ser mejorado para der una presentación más agradable.

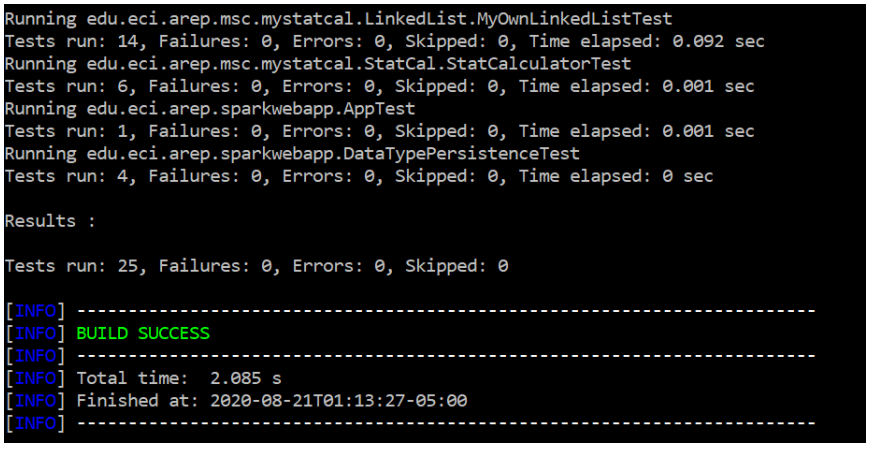


Figura 10: Pruebas Compiladas en MVN