Use Maven, HEROKU, GIT, Spark and GITHUB.

Santiago Martínez Martínez

Ingeniería de Sistemas
Noveno Semestre
Escuela Colombiana de Ingeniería Julio Garavito
Bogota DC - Colombia

Glosario

Java: Es un lenguaje de programación Orientado a objetos que hoy es parte de Oracle Corporation.

HTML: Es es lenguaje de marcado que nos permite clarar la estructura mediante etiquetas.

CSS: Hojas de estilo en cascada es un lenguaje que defina la apariencia de un documento escrito en lenguaje demarcado, permite que los usuarios personalicen los estilos de las etiquetas.

JavaScript: Es un lenguaje de programación que funciona en los navegadores de forma nativa.

Maven: Es una herramienta de manejo y automatización para proyectos de Java.

Spark: Es un framework para Java que te permitirá desarrollar aplicaciones web y se caracteriza por ser ligero y por tener una curva de aprendizaje favorable.

Promise: Es un objeto que representa la terminación o fracaso de una operación asíncrona

Resumen

Este documento tiene como objetivo explicar la implementación que se realizo para poder diseñar una aplicación web para cumplir los requerimientos solicitados.

En este taller se uso html para poder diseñar el esqueleto de nuestra pagina web, para poder hacerlo mas agradable a la vista del usuario utilizamos css que nos permitió acomodar a conveniencia la estructura de nuestro html.

Para poder interactuar con esta pagina se uso JavaScript, para el manejo del DOM y para los llamados asíncronos a los datos que solicitábamos al servidor.

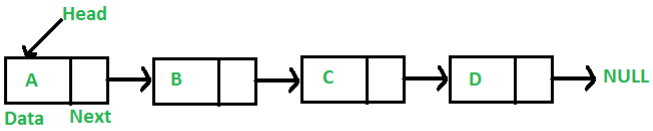
Por ultimo, utilizamos Java con el framework Spark el cual nos permitió poder responder las solicitudes que realizábamos con JS y con esto retornar la media y la desviación estándar teniendo en cuenta los datos ingresados por el usuario

Contents

1 Introduction

La construcción de este proyecto de realizo gracias a Maven, esta herramienta nos permitió dividir nuestros directorios del proyecto de una buena forma, para que puede ser extensible y entendible para los demás desarrolladores.

Lo que nos solicitan es poder generar una aplicación web que nos permita ingresar una serie de datos a los cuales le queremos calcular la media y la desviación estándar. Para almacenar estos datos utilice una implementación propia de una LinkedList, la cual nos ayudara a encapsular los datos en una "lista enlazada", para que estos datos puedan ser recorridos y usados se implementaron métodos que creemos necesarios para un buen funcionamiento. Estos métodos(add(), get(), size()) me permitió trabajar con la lista para solucionar el problema que nos solicitaban.



Para poder comunicar el front y el back fue necesario utilizar el lenguaje de programación JavaScript, gracias a sus funcionalidades por ser un lenguaje nativo para el navegador, nos permitió poder manejar los componentes de nuestro documento html y poder comunicar los datos ingresados por el usuario al servidor gracias a los llamados asíncronos que nos permite realizar.

2 Objetivos

El objetivo generar de este taller es aprender como desplegar de una manera correcta un proyecto mvn a Heroku y como trabajar con el cliente de heroku

para realizar cambios y poderlos correr de manera local.

Otro objetivo importante, es aprender como funciona el framwork Spark y como utilizarlo para poder realizar HTTP requests

3 Solución Requerimientos

Para la solución a los requerimientos se dividió el problema en 3 fases, la primera fase es el front de nuestra aplicación web, esta parte es la mas importante, ya que con ella interactuá nuestra solución con el usuario.

La segunda fase es la conexión entre el back y el front de nuestra aplicación, gracias a esta conexión comunicamos los datos que ingresa el usuario y las herramientas que creamos para hallar la media y la desviación en nuestro código.

La tercera y ultima fase de nuestra implementación fue la creación de nuestra aplicación web creada con ayuda de Spark, gracias a este leemos el JSON enviado por nuestra conexión, con este JSON, lo que hacemos es parsear los datos a Double y una vez ya tengamos esto devolvemos los datos de la media y la desviación estándar en la url que definimos.

4 Frontend

La interfaz con la que interactuar el usuario cuenta con 4 componentes esenciales, el primero es el titulo, el cual explica cual el propósito de la interfaz, el segundo es el input, en el cual el usuario agrega los números que desea ingresar separado por comas, el tercer componente es un boton que es sumamente importante ya que al oprimirlo, el activa un ActionListener el cual tiene como objetivo llamar a una función escrita en JavaScript que realiza el post al servicio que creamos con Spark, cuando spark devuelve la respuesta, esta es mostrada en el cuarto componente el cual muestra los resultados del calculo.

5 Backend

El backend, como ya comente antes fue escrito en Java con ayuda de Spark, lo que se hizo a grandes rasgos fue implementar un método POST a un url de nuestro servicio, el cual nos sirve de endpoint para conectarlo con el front, esta url fue 'calc', para esto al momento de realizar un POST a esta url, lo que hacemos es crear el objeto Statics, el cual tiene métodos estáticos que cumplen con las cualidades de la api de Java, que tienen como objetivo calcular la media y la desviación estandar, luego de crear este objeto, leer el JSON

```
<div class="container">
    <div class="formCal">
        <div class="title">
            <h1>Calculadora Estadistica</h1>
        </div>
        <div class="input">
            <input type="textfield" id="list" name=\"data\" value="" placeholder="ex: 10.0,15.0,22.0">
            <button class="button" type="submit" onClick="app.send()">Calcular</button>
        </div>
        <div class="result">
            <div class="row">
                <label for="mean">mean:</label>
                <output id="avg" />
            </div>
            <div class="row">
                <label for="stdv">stdv:</label>
                <output id="stdv" />
            </div>
        </div>
    </div>
</div>
```

teniendo en cuenta el parámetro request que tiene nuestra funcion lambda de Spark y con esto la parseamos para que nuestro programa pueda hacer los calculos, acabado esto retornamos la respuesta anidando las a un nombre de respuesta que luego con JavaScript atraparemos.

```
Run | Debug
public static void main(String[] args){
    port(getPort());
    staticFiles.location("/public");
    post("/calc", (request, response) -> {

        Statics statics = new Statics();
        statics.readJSON(request.body());

        return "{\"media\":" + statics.mean() + ", \"desviacion\":" + statics.standardDesviation() + "}";
    });

}

static int getPort(){
    if (System.getenv("PORT") != null) {
        return Integer.parseInt(System.getenv("PORT"));
    }
    return 4567;
}
```

6 Conexion

La conexión es el factor fundamental para que todo el sistema funcione, para poder implementarlo se realizo una promise, que en palabras simples, es una operación la cual nos retorna dos respuesta de un Request HTTP, por un lado, si la solicitud que realizamos con la promesa es correcta, nos devolverá la información o recurso que solicitamos por lo contrario, nos retornara un error. luego de tener la respuestas que es la data que nos retorno el backend, con

JQuery que es un framework de JavaScript el cual nos ayuda a manejar los elementos del DOM, reasignamos en forma de texto teniendo en cuenta los nombres de la respuesta que indicamos en el backend(media, desviación) a los div que creamos para la respuesta, los cuales deben de tener un id para que JQuery los identifique.

```
const numbers = (error, data) => {
    if (error != null) {
        console.log(`Error: ${error}`);
        alert("Formato Invalido -> ej: 10.5,3.5,5");
        return;
    }
    $("#avg").text(data.media);
    $("#stdv").text(data.desviacion);
}

const send = () => {
    var data = $("#list").val();
    var listNum = data.split(",");
    postMethod(listNum, numbers);
}

const postMethod = (listNum,numbers) => {
    var promise = $.post({
        url: "/calc",
        data: JSON.stringify(listNum),
        contentType: "application/json"
    });
    promise.then((data) => {
        numbers(null, JSON.parse(data));
    },(error) => {
        numbers(error, null);
    });
}

return {
    send: function() {
        send();
    }
}
```

7 Pruebas

Las pruebas implementadas en este programa van dirigidas a los métodos que calculan la media y la desviación estándar.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.escuelaing.arep.app.AppTest
15.33
Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.054 sec

Results :

Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
```

8 Conclusion

1. Se noto en este laboratorio que para desarrollar una aplicación web es necesario de tener conocimiento aunque sea mínimo en múltiples tecnologías.

2. La ayuda de frameworks hace que el desarrollo de la solución sea mas facil de implementar ya que nos suministran metodos y herramientas que ya estan listas para utilizar.

3. Los canales de conexión entre las diferentes componentes del sistema son de suma importancia, ya que gracias a estos podemos transferir datos, en nuestro caso, en formato JSON.

References

[1] *MDV.*
`https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_`$global es/Promise$

[2] *Promises Chaining.*
`https://javascript.info/promise-chaining`

[3] *Documentation.*
`https://sparkjava.com/documentation`

[4] *Cómo crear un API con Spark Java.*
`https://platzi.com/blog/api-spark-java/`

[5] *Tutorial.*
`https://es.overleaf.com/learn/latex/Tutorials`