Taller Clientes y Servicios

Calderón Ortega Andrés Mateo

Arquitecturas Empresariales
Ingeniería de sistemas, Escuela Colombiana de Ingeniería Julio Garavito, Bogota, Colombia
13 de febrero de 2021

Resumen: Se realizará la explicación del funcionamiento de un framework experimental que intenta ser similar al framework Spark, el desarrollo guiado a una programación orientada a objetos, su arquitectura básica, la forma en la que se decide leer los datos y archivos y la conexión con la base de datos elegida, en este caso PostgreSQL.

1. Introducción

El Problema tratado en este proyecto es el de la implementación de un framework web que sea capaz de ofrecer servicios similares al framework Spark, este nuevo 'mini' framework deberá ser capaz de resolver peticiones get por medio de funciones lambda y permitir el acceso a recursos estáticos tales como: Páginas HTML, Imágenes, y archivos de tipo JavaScript y CSS, para probar que el framework funcione de manera correcta, el usuario deberá ser capaz de ingresar datos a una base de datos (en este proyecto PostgreSQL) y leer los datos de la misma, en un ambiente intuitivo para él, la razón de este desarrollo de un mi framework es por las desventajas que trae para una organización el uso de framewrok 'grandes' como Spark; esto debido a que el uso de un framework como Spark, toma mucho más tiempo de aprendizaje respecto a un framework 'mini' creado desde cero, además de que estos pueden tener versiones muy inestables, causan un menor rendimiento debido a la gran cantidad de funcionalidades y extras que traen, y por supuesto quedan librerías y código inutilizable y des aprovechado a diferencia de un framework 'mini' que se implementa exclusivamente para las necesidades de la organización.

Para este proyecto se utilizó la versión 11 de Java; es de vital importancia para el correcto funcionamiento del mismo que esta esté instalada en la máquina donde este sea ejecutado, adicionalmente fue construido haciendo uso del ambiente de desarrollo IntelliJ y para el manejo de dependencias se utilizó Maven.

2. Arquitectura

2.1. Diseño Http

Para la solución de este problema creamos la clase HttpServer que será la encargada de procesar las peticiones y manejar el puerto por el cual el servicio correrá, para esto se hace uso de la librería 'java.net' la cual nos dará la clase ServerSocket, inicialmente abriremos un puerto haciendo uso de este socket y mantendremos un ciclo infinito para escuchar y atender las peticiones de manera secuencial, una vez tenemos una clase lista para atender las peticiones debemos crear las clases que simularan al framework Spark, para esto creamos las clases 'PicoSpark' y 'PicoSparkServer'.

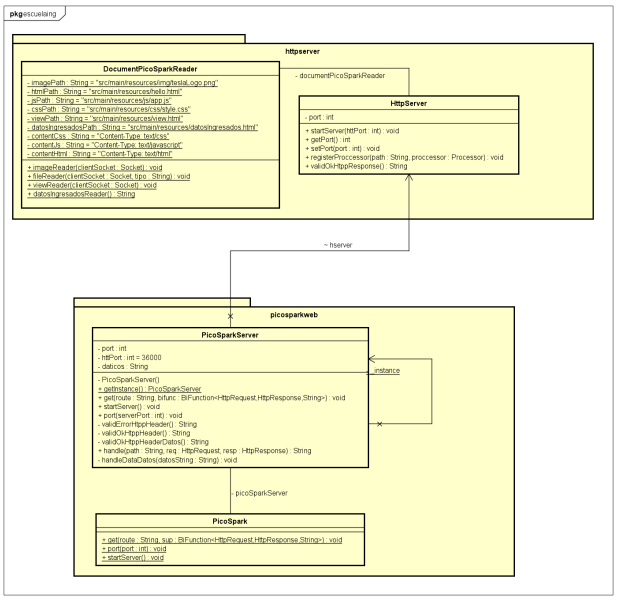

Figura 1: Diagrama De Clases 1

La clase 'PicoSpark' será la encargada de entender las peticiones get por medio de funciones lambda las cuales le indicaran el recurso y el 'path' por el cual desea atender, está una vez llega la petición get pedira la instancia de la clase 'PicoSparkServer', y delegará la tarea de la petición get, esta también será la encargada de designar el puerto por el cual se va a correr y adicionalmente es la que iniciara el servidor 'PicoSparkServer', este servidor está programado para permitir un flujo correcto después en el path '/Apps' cualquier petición que no empiece con '/Apps' el servidor se encargara de responder con un mensaje que dice 'Dirección No Válida, Use /Apps' y si está dentro del path '/Apps' pero no corresponde a alguno de los get permitidos responderá con un mensaje que dice 'I Found and Error', todas estas peticiones será registradas para que el HttpServer las maneje de la manera indicada.

Para el manejo de peticiones y la resolución de archivos se implementó una clase que llama 'DocumentPicoSparkReader' que se encargara de escribir en el socket haciendo uso del OutputStream el archivo solicitado además de colocar el encabezado pertinente para cada tipo de archivo (HTML, JavaScript, CSS y jpeg), adicionalmente si debe responder con los datos de la base de datos este realizara la petición a la clase encargada de manejar la conexión con la base de datos, la cual le retornara la data para ser escrita en el archivo de visualización de los datos.

2.2. Diseño Base De Datos

Para solucionar el problema de la inserción y consulta sobre una base de datos, se decidió utilizar la base de datos PostgreSQL que nos ofrece Heroku, en este caso se creó la clase 'DataBaseConnectionJDBC' que se encargara de realizar la conexión con la base de datos remota mediante el uso del driver 'org.postgresql.Driver' y retornara la conexión a la clase 'DataBaseJDBC' que se encargara de realizar las peticiones de inserción y consulta de los datos, esta base de datos almacena una lista de datos y la suma de los mismos, para la resolución del problema se utiliza el Java Database Connectivity que es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

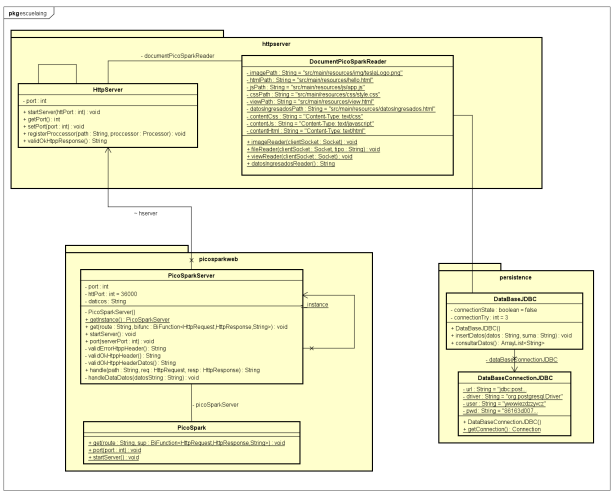

Figura 2: Diagrama De Clases 2

2.3. Diseño General

Para probar el correcto funcionamiento del framework y de la aplicación en general, reutilizaremos la implementación de una calculadora estadística la cual esta vez se conectara a la base de datos para guardar la información y será la encargada también de realizar la suma y conversión de los datos, además tendremos la clase 'DemoRunTime' la cual implementara el 'mini' framework recien realizado, permitiendo las peticiones a dos HTML, un JavaScript, una imagen y un archivo CSS

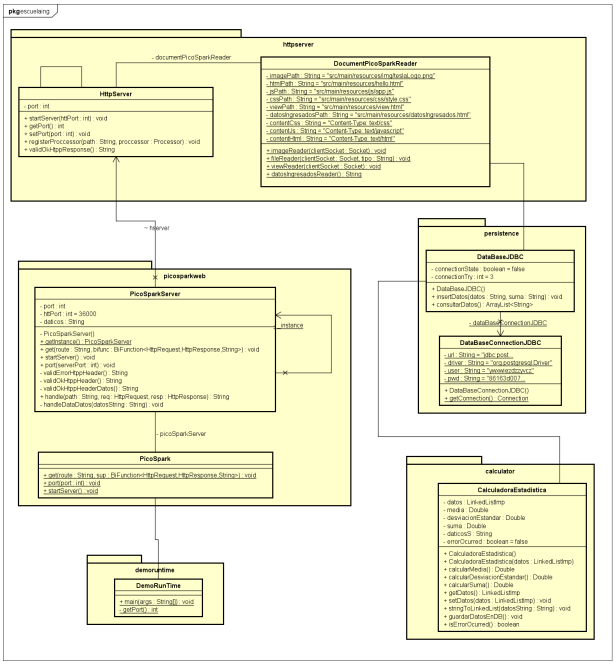

Figura 3: Diagrama De Clases 3

3. Despliegue y Construcción

La construcción de la aplicación se da haciendo uso de la JVM número 11 por la facilidad que nos brinda al momento de utilizar el paquete 'java.net.http' y con este las clases 'HttpRequest' y 'HttpResponse', como ambiente de desarrollo utilizamos IntelliJ IDEA en su versión Ultimate que nos permite un mejor manejo de archivos css, js y java, dándonos así un plus a la hora de desarrollar adicionalmente decidimos utilizar el manejador de dependencias Maven para tener una integración completa entre todos los componentes.

Figura 4: Pagina Principal

Para el despliegue de la aplicación utilizamos Heroku como servidor, debido a su gran integración con git, github y java, para esto debimos modificar el archivo '.propierties' para decirle a Heroku que corra con la máquina 11 de Java.

Figura 5: Ingreso de Datos

4. Conclusiones

Se adquirieron conocimientos en el uso de Sockets en Java para la construcción de servidores Http, además se pudo reflejar y conocer el parcial funcionamiento interno de frameworks web tales como spark, también se pudo ver las ventajas que tiene el construir un framework desde cero debido a la velocidad de procesamiento que este ofrece al momento de implementarse, pero sobre todo se vio el gran trabajo y la dificultad que conlleva la construcción de este tipo de frameworks y la complejidad de los mismos y con esto se pudo ver el porqué hoy en día es más común utilizar frameworks ya construidos como spark, debido a la agilidad y rapidez que estos ofrecen en el desarrollo, además que estos minimizan errores y de haberlos hay una mayor facilidad para solucionarlos.

Referencias

[1] tiThink Technology. 2021. Framework o librerías: ventajas y desventajas. [online] Available at: https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/ [Accessed 13 February 2021].].

[2] host, o., Bhatraju, S. and Taherian, P., 2021. org.postgresql.util.PSQLException: FATAL: no pg hba.conf entry for host. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/25641047/org-postgresql-util-psqlexception-fatal-no-pg-hba-conf-entry-for-host [Accessed 13 February 2021].

[3] Journals.ecs.soton.ac.uk. 2021. Reading from and Writing to a Socket. [online] Available at: http://journals.ecs.soton.ac.uk/java/tutorial/networking/sockets/readingWriting.html [Accessed 9 February 2021].

[4] Daniel Benavides. 2021. Clase de laboratorio. [5 Febrero 2021].