Artículo # 1 AREP

First A. Author, Cesar Eduardo Lanos Camacho, PROYECTO #1

1. **Introducción**

En este documento se hablará sobre cómo fue la construcción de un servidor Web (tipo Apache) en Java. El servidor debe ser capaz de entregar páginas HTML e imágenes tipo PNG, como también pojos para que se pueda en un futuro crear un futuro un framework.

Primero se expondrá cómo ejecutar el código y se presentará también el diseño general y arquitectura del código y el porqué de la implementación, luego se presentarán los resultados de las pruebas que se le hicieron al código probando todo su funcionamiento y también probando casos en los que el servidor botaría un error en particular y por último se expondrá las conclusiones sobre la realización de este proyecto. También se utilizó una herramienta que permitió montar y tener nuestra aplicación alojada en la web llamada HEROKU y el link del aplicativo es el siguiente:
https://proyectoarem1.herokuapp.com/index.html (podrá ver en funcionamiento la aplicación vía web sin necesidad de descargar el proyecto y correrlo localmente).

2. **PASOS PARA LA PREPARACIÓN Y EJECUCIÓN DEL CÓDIGO**

a. Visite el repositorio del proyecto del siguiente enlace https://github.com/arem2019-1/Proyecto1Arem copie el link SSH del proyecto y usando Linux haga una clonación en su carpeta preferida o copie la siguiente línea de código y péguela en su consola para alojar el proyecto:

git clone https://github.com/arem2019-1/Proyecto1Arem

b. Una vez ubicado en la raíz del proyecto, ejecutar el siguiente comando en la terminal:

mvn package

c. Si requiere la documentación del proyecto (JAVADOC), por favor generala con el siguiente comando:

mvn javadoc:javadoc

d. compile el proyecto en terminal desde la carpeta raíz ejecutando la siguiente línea:

java-cp target/WebServerProeyctArem-1.0-SNAPSHOT-jar-with-dependencies.jar co.edu.escuelaing.arep.HttpWebServer

una vez compilado el programa se ejecuta en el puerto 4567, para probarlo vaya a esta dirección desde su navegador preferido:

i. para ver el HTML:

http://localhost:4567/index.html

ii. para ver sólo la imagen PNG:

http://localhost:4567/imagen.png

iii. Para ver el uso de los pojos.

http://localhost:4567/fram/cuadrado/78

o

http://localhost:4567/fram/suma/78

3. **CONTEXTO DEL APLICATIVO**

Para este proyecto se consideró a solucionar según los requerimientos de nuestro profesor que eran los siguientes:

A. Se deberá construir un servidor Web (tipo Apache) en Java.

1. Modelo hecho en la web https://www.lucidchart.com/
2. Pantallazo Prueba 1

B. El servidor debe ser capaz de entregar páginas html e imágenes tipo PNG.
C. Usando el servidor se debe construir un sitio Web de ejemplo y desplegarlo en Heroku.
D. El servidor debe atender múltiples solicitudes no concurrentes.
E. De proveer un framework IoC para la construcción de aplicaciones web a partir de POJOS

4. DISEÑO DEL APLICATIVO

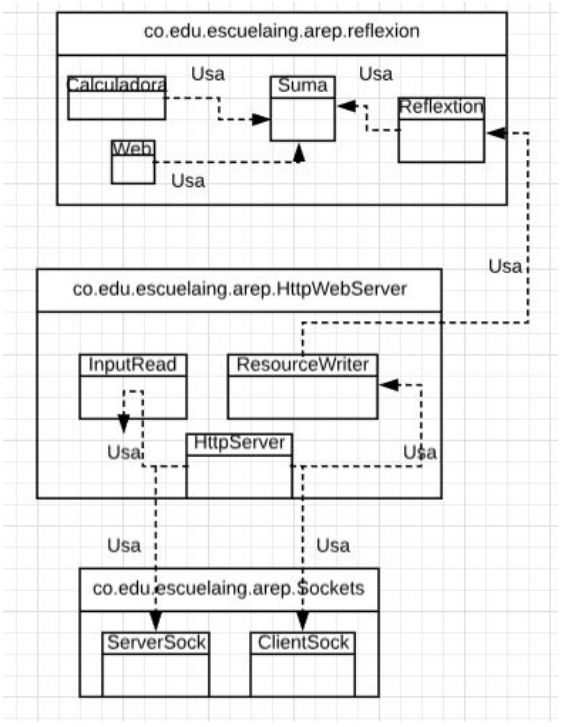

Figura 1

Modelo representativo tipo MVC (Modelo Vista Controlador). Usando conceptos básicos de modularidad por medio de clientes y servicios. Patrón utilizado Patrón 1: Remote procedure call (RPC), Aunque todo se hace localmente, pero se podría decir que los recursos alojados son una mini base de datos alojada también localmente.
Como se aprecia en el modelo de la figura 1, se cuenta con 5 clases java donde la clase principal (Main) es HttpServer.
HttpServer usa a ServerSocket, ClientSocket, inputReader y ResourceWrite.
InputReader se encargará de procesar la información importante de la solicitud para que ResourceWrite en base a esa respuesta se encargue del envió según la solicitud del InputReader.
ServerSocket se encarga de crear y retornar el socket necesario para la recepción de solicitudes (Servidor) y ClientSocket para en el envió de esas respuestas.

Y a su vez ResourceWriter usa la clase Reflection, que a su vez usa la interfaz web, de la que implementan otras dos clases, Calculadora y Suma, y estos vienen siendo los dos pojos de la aplicación. Con estos el usuario al mandarle una de las palabras clave puede hacer una determinada tarea que saldrá en el navegador.

5. Pruebas

Se presentarán a continuación xxxx pruebas diferentes del aplicativo:

a. Probando un .PNG

. Visualizando en Heroku
https://proyectoarem1.herokuapp.com/bug.png

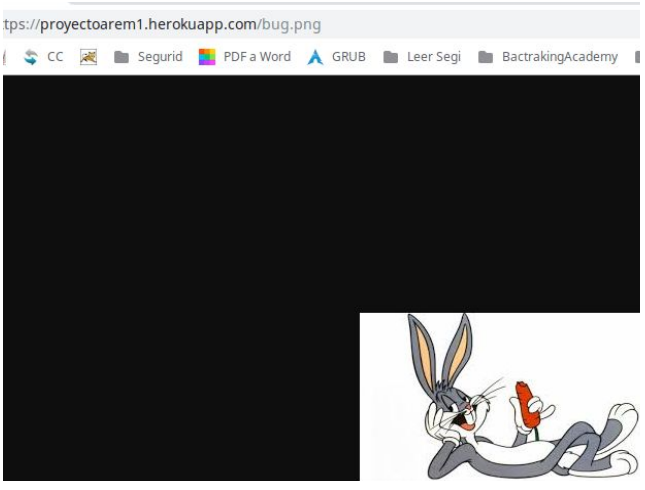

Figura 2

Aquí simplemente el servidor en funcionamiento localmente se prosigue a abrir un archivo imagen en este caso IMAGEN.PNG, se hace la solicitud al navegador ingresando a http://localhost:4567/bug.png por consiguiente

1. Modelo hecho en la web https://www.lucidchart.com/
2.Pantallazo Prueba 1

la respuesta que nos manda es el archivo que está previamente creado y
alojado en la carpeta de resources sin ningún problema alguno.imagen

b. Probando un HTML

Figura 3

i. Visualización Heroku

https://proyectoarem1.herokuapp.com/index.html

Aquí simplemente el servidor en funcionamiento localmente se prosigue a abrir un archivo .HTML en este caso index.html, se hace la solicitud al navegador ingresando a http://localhost:4567/index.html por consiguiente la respuesta que nos manda es el archivo que está previamente creado y alojado en la carpeta de resources sin ningún problema alguno.

c. Probando POJO cuadrado

i. Visualización Heroku

https://proyectoarem1.herokuapp.com/fram/cuadrado/10

Figura 4

Aquí mostramos como funciona uno de los POJOS y el servidor provee un framework con el cual se pueden construir aplicaciones a partir de él.

Su función es sencilla, en la url se pasa /fram/cuadrado/# y esto nos regresara el cuadrado del número que le hayamos pasado(#=número).

d. Probando POJO suma

i. Visualización Heroku

https://proyectoarem1.herokuapp.com/fram/suma/10

Figura 5

Aquí mostramos como funciona uno de los POJOS y el servidor provee un framework con el cual se pueden construir aplicaciones a partir de él.

Su función es sencilla, en la url se pasa /fram/suma/# y esto nos regresara el la suma del número que le hayamos pasado(#=número).

6. Conclusiones

Referencias

Luis Daniel Benavides Navarro, “Integración empresarial” Diapositiva. Ciudad de Publicación, (only Bogotá), Ciudad: Colombia.

Jorge V. (2011). Sockets en Java (cliente y servidor): Codigoprogramacion. Recuperado de http://codigoprogramacion.com/cursos/java/103-sockets-en-java-con-cliente-y-servidor.html#.W5W3YM5KjIUNA. (2017). Java Socket Server Examples (TCP/IP): codejava.Recuperado de https://www.codejava.net/java-se/networking/java-socket-server-examples-tcp-ip.

1. Modelo hecho en la web https://www.lucidchart.com/
2.Pantallazo Prueba 1