Servidor web concurrente y cliente en AWS (Septiembre 2019)

Yohanna Andrea Toro Duran

I. INTRODUCCIÓN

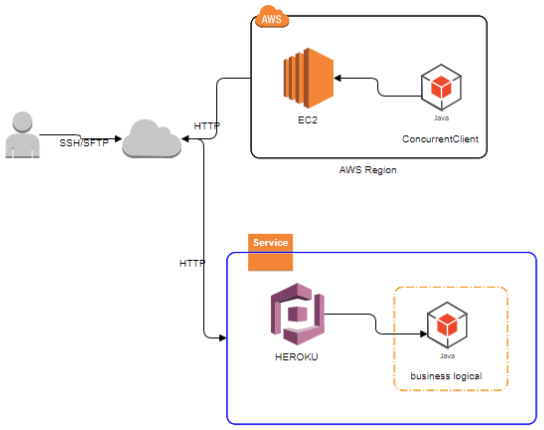
En la arquitectura de software existe una serie de decisiones sobre que patrón elegir al momento de crea una arquitectura, en este caso vamos a resaltar el patrón cliente-servidor.

Algunos de los beneficios que nos trae el uso de este patrón:

1. Crear sistemas modulares.
2. Tolerante a fallas.
3. Sistemas seguros.
4. Facil mantenimiento.
5. Escalabilidad.

Para ello se elaboró un servidor web concurrente tipo apache que realiza una respuesta en forma de html o imagen png, adicional a lo mencionado se realizó una construcción de un framework IoC a partir de POJOs
Desde el lado del cliente se creo una instancia en aws, en el cual se crea un pool de hilos que se encargan de realizar múltiples peticiones al servidor. Esto se hace con el fin de determinar cómo es capaza de responder un servidor ante múltiples peticiones que se realizan a la vez, pero también

II. ARQUITECTURA DE SOFTWARE

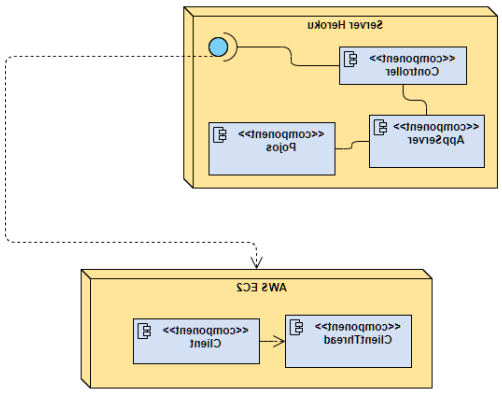


Presentación: se encuentre la aplicación web que es la interfaz con la que el usuario interactúa, esta aplicación web se encuentra desplegada en en una instancia EC2 en AWS que realiza múltiples peticiones solicitando un recurso ya sea html, png o algún metodo

• Servicio: En la capa de servicio se encuentra el frameworks y el servidor web concurrente , este llama al

controlador y crea un pool de hilos y se conecta con la aplicación que brinda el servicio que da respuesta de un método o un recurso, dependiendo de la solicitud.

ARQUITECTURA DE DESPLIEGUE



El despliegue del servicio web concurrente se realizó en heroku según los requerimientos establecido, este servicio se escucha por el puerto 5000 y el cliente se encuentre desplegado en una maquina de AWS este cliente realiza las peticiones mediante HTTP a heroku el cual le va responder dependiendo de el numero de peticiones

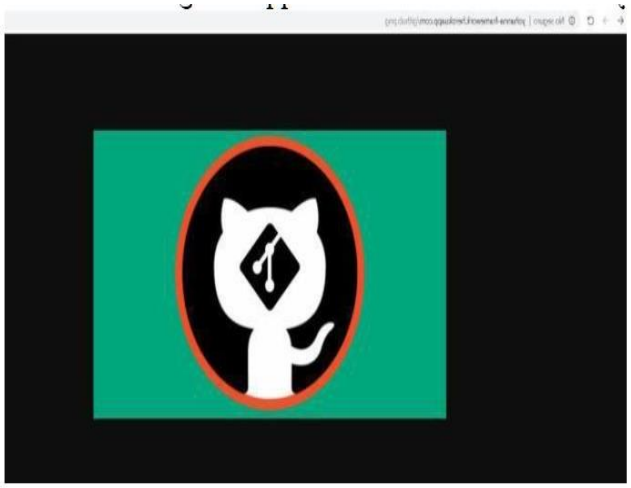
IV. APLICACION WEB

La aplicación web se desplego en heroku la cual recibe peticiones por el puerto 5000, esta aplicación esta construida con Maven y uso de POJOs
Peticiones que recibe

- Html
- Png
- test
- hola:name

casos de prueba que se usaron:

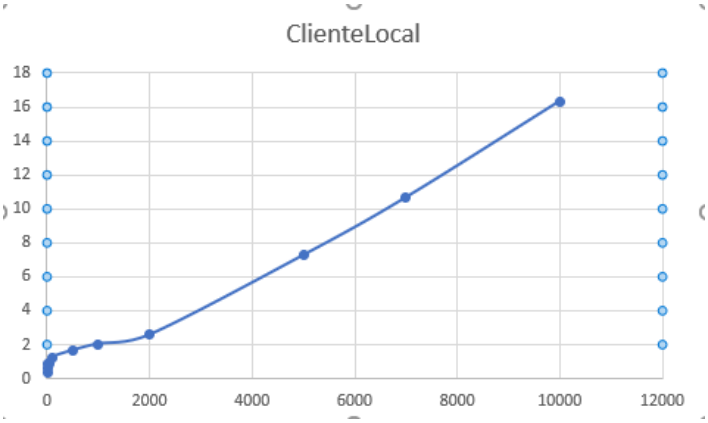
- yohanna-framework.herokuapp.com/apps/test



PRUEBAS

Para determinar se realizó una toma de 10 datos que tiene dos variables e el eje x corresponde el número de solicitudes que realiza el cliente AWS y la variable correspondiente al eje y son es el tiempo en segundos que se demoró en responder el servidor

el mismo proceso se realizó pero en vez de tener el cliente funcionando desde AWS se realizó de forma local, cabe aclara que en ambas muestras el servidor se encuentra desplegado en heroku



Se puede observar que en ambas muestras las gráficas tienen la misma grafica pero lo que marca la diferencia es que de forma local el servidor es mucho más rápido en responder mientras que el cliente AWS se demora mucho más en responder

CONCLUSIÓN

Podemos concluir que el uso de anotaciones POJOs nos permite entender como esta estructurado un servidor web y como a travez de estas anotaciones podemos realizar peticiones http. Ya con un servidor web se pudo realizar el correspondiente framework que nos permitió entender como funcionan desde las funciones mas básicas como es inicializar las clases hasta generar la correspondiente solicitud pedida por el usuario. Adicional a esto también podemos concluir que tener un modelo cliente servidor trae beneficios ya que permite que un diferentes clientes pidan solicitud a un mismo servidor y como este interactúa para dar respuesta pero también trae desventajas ya que al realizar la solicitud a un mismo servidor se encuentra un punto de error que es que en el momento en que falle el servidor los demás clientes no podrán realizar ninguna solicitud, por ende es bueno el modelo pero también sería bueno usar una distribución de carga para evitar fallos o posibles errores en un solo punto.