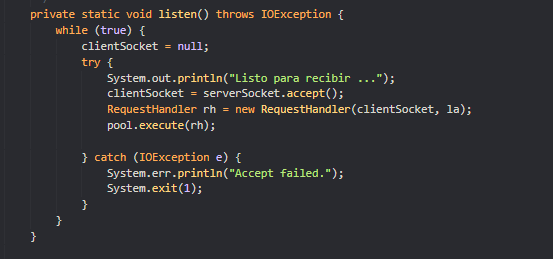
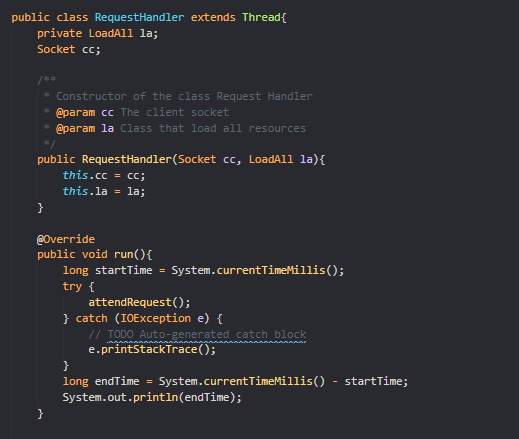
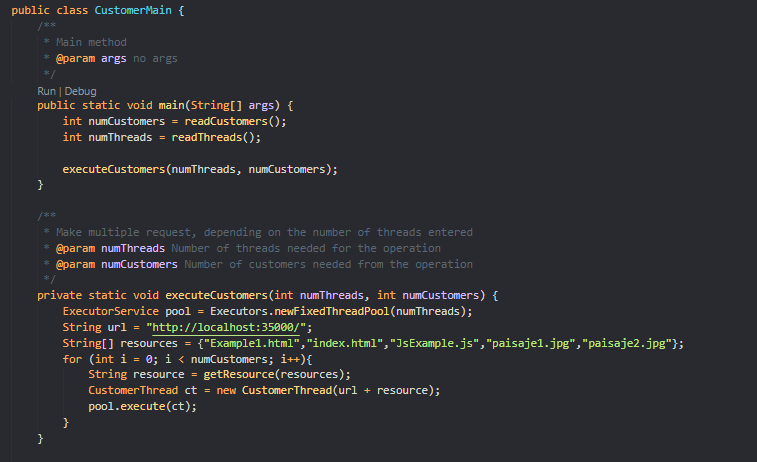
Evidencia del trabajo en clase.



La imagen anterior, muestra el método que realiza la paralelización del servidor, hace uso de un threadpool, al cual se le pueden especificar la cantidad de hilos que se desean ejecutar a la vez, este proceso se encuentra dentro de un ciclo while, el cual permite ejecutar un número indefinido de peticiones.



La imagen anterior, muestra el hilo que se encarga de atender cada petición al servidor, el método run, se ejecuta una vez que se le de la orden desde la función principal, este a su vez invoca un método, el cual ya se había implementado antes, que se encarga de atender la petición del cliente.



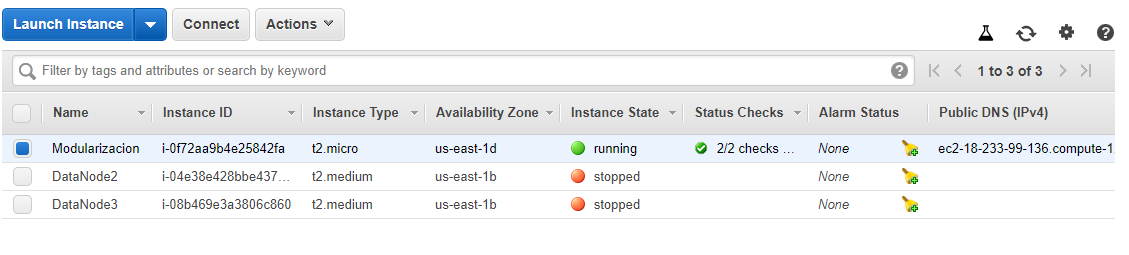
Por último, se implementó un cliente, el cual también es paralelizado de manera similar a como se hizo con el servidor web, pues desde una función principal, se crea una cantidad definida de hilos que realizan peticiones simultaneas al servidor que se había creado antes.



Aquí tenemos la clase que extiende de thread, el cual se encarga de realizar la petición al servidor web.

**Enunciado de Tarea:**

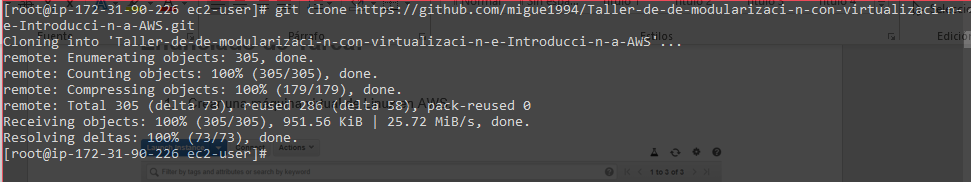
1. Crear una máquina virtual de Linux en AWS.



Se creó la máquina virtual en AWS educate, según los pasos indicados por el profesor, esta máquina es la que tiene el nombre de Modularización.

1. Desplegar un cliente que lea URLs e imprima contenidos en consola, en la máquina virtual de AWS.

Para transferir los archivos del cliente, debemos instalar Maven, git y java en nuestra máquina de aws, posteriormente, solo clonamos el repositorio que está alojado en github, dentro de la máquina remota.

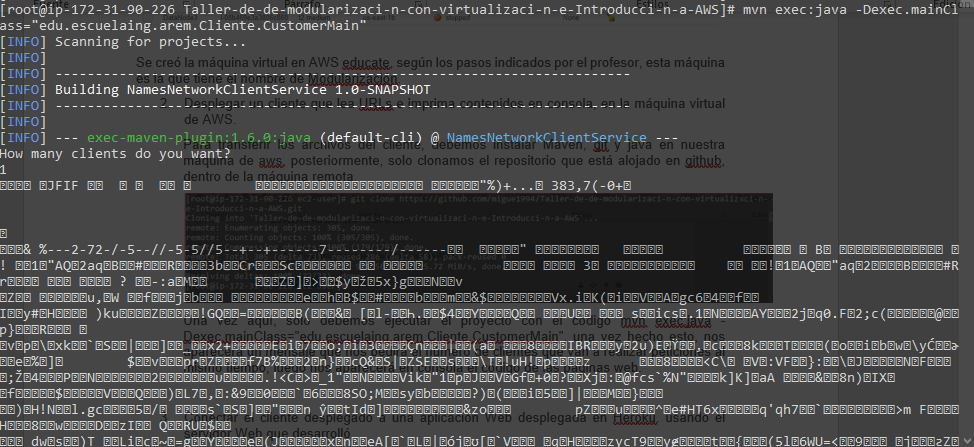


Ejecutamos el proyecto, el cliente por defecto solo tiene una url, que es la que apunta a heroku, por lo tanto, nos saldrá un código almacenado dentro del servidor, sin embargo, con esto se garantiza que el cliente funciona para cualquier Url disponible



1. Conectar el cliente desplegado a una aplicación Web desplegada en Heroku. usando el servidor Web que desarrolló

Una vez aquí, solo debemos ejecutar el proyecto con el código mvn exec:java -Dexec.mainClass="edu.escuelaing.arem.Cliente.CustomerMain", una vez hecho esto, nos aparecerá un mensaje que nos pedirá el número de clientes que van a realizar peticiones al mismo tiempo, luego nos aparecerá en consola el código de las páginas web.



El servidor nos responde con caracteres inentendibles, debido a que está solicitando una imagen como recurso, y como tal esta no puede ser enviada por consola.

1. Modifique su servidor Web para que responda múltiples peticiones de manera concurrente.

Este procedimiento se realizó en clase, y se explicó en la primera parte de este documento

1. Modifique su cliente para probar su servidor web haciendo muchas peticiones en paralelo.

Este procedimiento se realizó en clase, y se explicó en la primera parte de este documento

1. Mida el desempeño de su servidor web variando el número de hilos que soporta y aumentando o disminuyendo el número de peticiones concurrentes.

Al realizar 100 peticiones al servidor de manera concurrente, el responde de adecuadamente, sin presentar alguna interferencia o bloqueo, sin embargo, al intentar realizar el mismo experimento con 1000 peticiones, el servidor que esta desplegado en heroku, no es capaz de manejar tantas al mismo tiempo, y debe realizar una espera, posteriormente, sigue respondiendo peticiones hasta donde puede, este proceso se tarda un poco, debido a las esperas.

1. Pruebe tanto contenidos estáticos como dinámicos.