**Daniel Augusto Ramírez Dueñas – 201513523**

**Juan Francisco Gutiérrez Amin – 201511872**

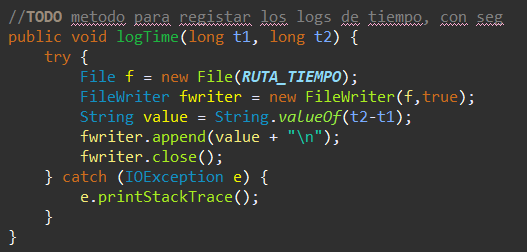
**Análisis Caso 3**

1. **Descripción detallada de la implementación de los monitores**

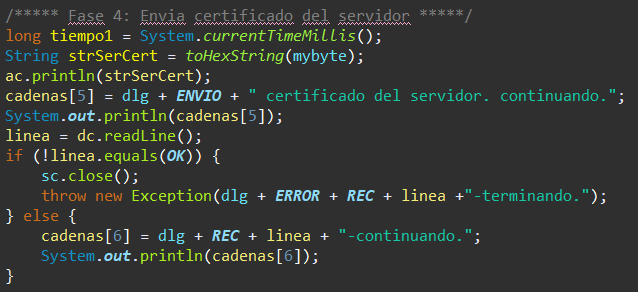
Partiendo del proyecto realizado para el caso 2, tomamos las clases de Cliente, Simétrico y Asimétrico para la implementación del sistema Time & Attendance. En este caso, el objetivo es evaluar el desempeño de la aplicación con distintas pruebas de carga. Estas pruebas tienen en cuenta cuatro distintos aspectos: número de threads, número de transacciones, retardos de tiempo y la implementación de cifrado o ausencia de este.

1. **Con seguridad**

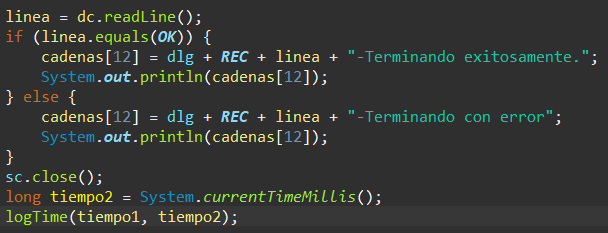
Inicialmente, para el registro del tiempo, creamos un log en formato .csv. Para ello, iniciamos en la clase D, en la cual agregamos un método que se encarga de crear el archivo.



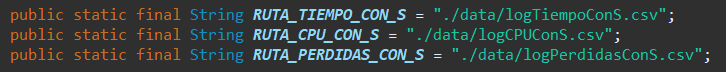
Luego, al iniciar la fase 4 de esta clase, es decir, antes de enviar el certificado del servidor, se inicia el conteo del tiempo.

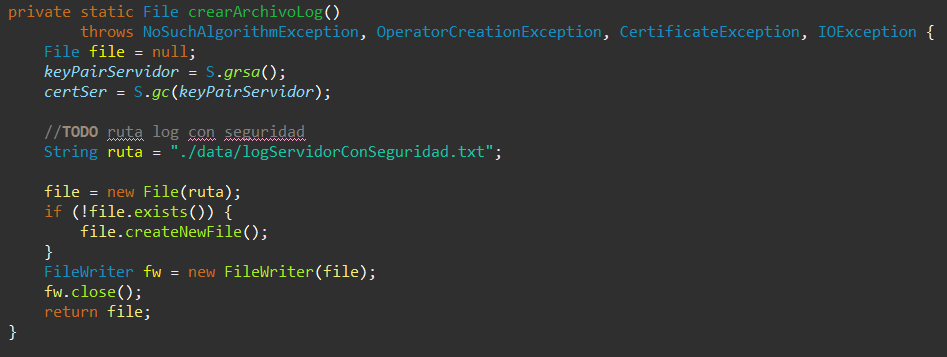


Finalmente, se registra el tiempo al finalizar la comunicación con el servidor.

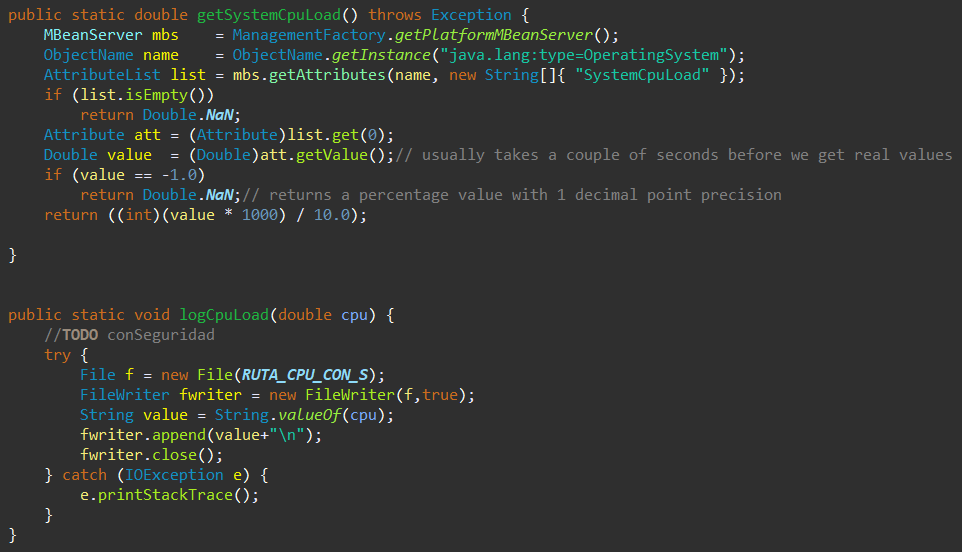


Por otro lado, En la clase C, manejamos los logs de tiempo, uso de CPU, transacciones perdidas, y todas las acciones que se realizan.

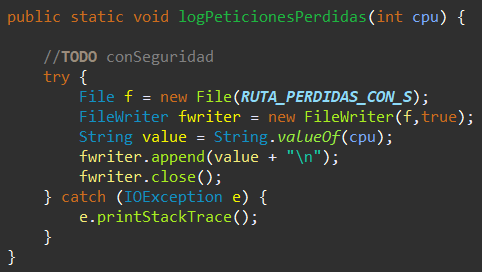




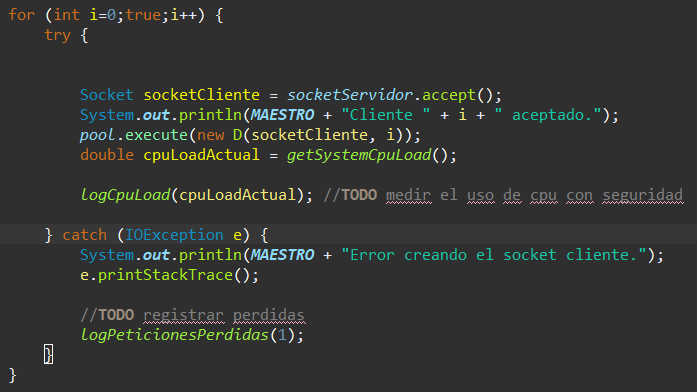
En el caso del log de CPU, utilizamos los siguientes métodos.



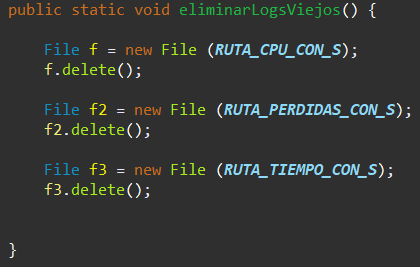
Para el de transacciones perdidas, se utiliza el siguiente método.



Finalmente, ambos se utilizan al correr el servidor y comunicarse con el cliente en el main.



Por otro lado, también se encarga de borrar todos los logs, cada vez que se entabla una nueva comunicación cliente – servidor.

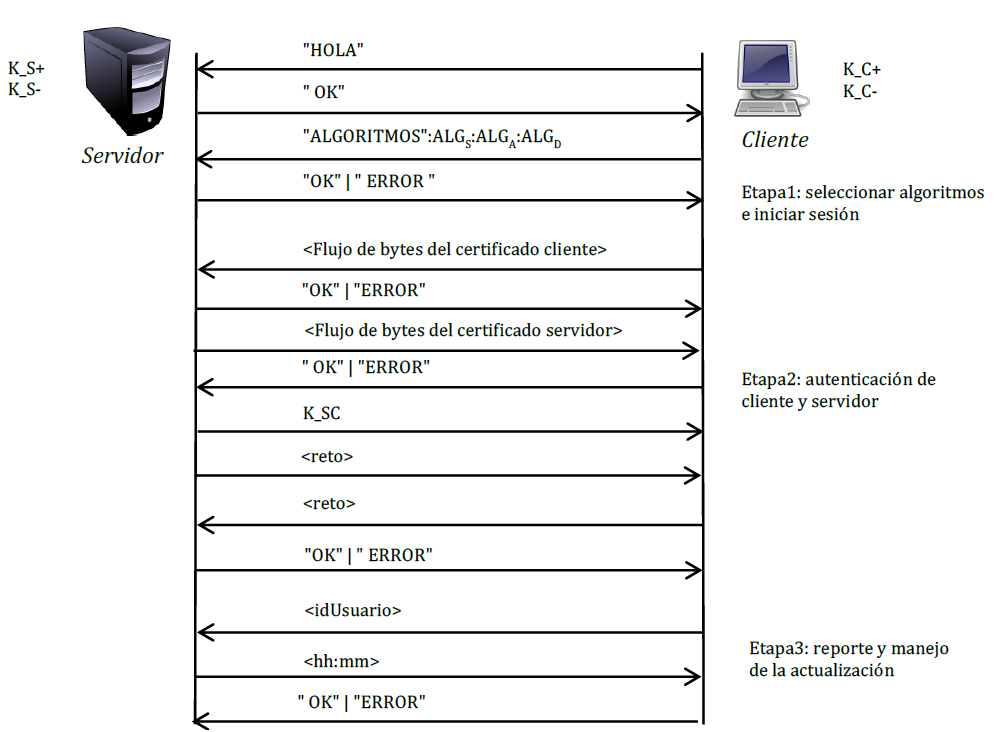


Finalmente, para crear la cantidad de transacciones y retardo de tiempo requeridas en cada escenario, utilizamos la clase Generator, la cual se encarga de cumplir con dichos requerimientos por medio de la herramienta GLoad.



1. **Sin seguridad**

Para este caso, se cambia únicamente el protocolo del cliente. Las demás clases son iguales a “Con seguridad”, exceptuando la ausencia de Simétrico y Asimétrico, ya que no se utilizan en esta ocasión. Dicho protocolo es el siguiente:



1. **Identificación de la plataforma**

**Arquitectura:** 64 bits

**Numero de núcleos:** 4

**Velocidad del procesador:** 1.50 GHz

**Tamaño de la memoria RAM:** 8 GB

**Espacio de memoria asignado a la JVM:** 124.780544 MB

1. **Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia**
2. **Tiempo transacción**





1. **Número de transacciones perdidas**





1. **Porcentaje de uso de CPU**





**Conclusiones:**

* A mayor tiempo de retardo, mayor tiempo promedio por transacción y menor porcentaje de uso de CPU.
* Para una misma carga, tarda más en realizar una transacción si se tienen más threads.
* Se puede observar que los datos obtenidos para el uso de CPU son fiables, debido a la baja varianza y desviación estándar que presentan.
* Se puede observar que los datos obtenidos para el tiempo de transacción son fiables, debido a la baja varianza y desviación estándar que presentan.
* Debido a la plataforma utilizada para realizar las pruebas, no se obtuvieron transacciones perdidas en ninguno de los casos.

1. **Comportamiento de la aplicación ante diferentes niveles de seguridad**





**¿Cuál es el resultado esperado sobre el comportamiento de una aplicación que implemente funciones de seguridad vs una aplicación que no implementa funciones de seguridad?**

El resultado esperado es que, si se trata de una aplicación con seguridad, el comportamiento de la aplicación va a requerir mayor uso de CPU, debido a que, al usar métodos de cifrado, la aplicación necesitará hacer una mayor cantidad de operaciones, lo cual elevaría el uso de la CPU, mientras que una aplicación que no maneje nada de seguridad usará menor cantidad de CPU.

**Conclusiones:**

* A pesar de que se obtuvieron menores tiempos por transacción en el caso sin seguridad, no se obtuvieron los resultados esperados, ya que la cantidad de transacciones y tiempo de retardo era la misma. No obstante, aún siendo un menor número de operaciones por la falta de cifrado y descifrado, esto solo se ve reflejado en el tiempo de operación, más no en el uso de CPU.