**Pre-experimentación:**

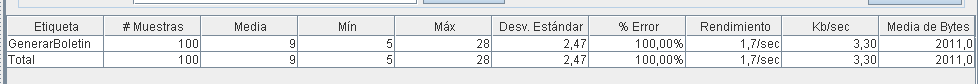
* **Problemática:** Se plantea desarrollar una arquitectura por capas que use servicios REST para la comunicación con los sensores por medio de peticiones HTTP y que sea capaz de cumplir con requerimientos funcionales y no funcionales de la aplicación.
* **Objetivo del experimento:** Probar la arquitectura planteada para verificar el cumplimiento de los requerimientos no funcionales
* **Descripción del experimento:** Se desarrollará la capa de persistencia, la capa de presentación y se expondrán nuevos servicios con el fin de suplir otros requerimientos funcionales. Por medio de pruebas de carga se probará la latencia y escalabilidad del software.
* **Artefactos a construir:** Se construirá la capa de persistencia dentro del backend del proyecto y en otra capa se realizará la interfaz gráfica para que desde allí se consuman los recursos de la aplicación.
* **Recursos de la experimentación:** El experimento se desarrollará en el ambiente de ejecución NetBeans 8.1, como servidor para desplegar la aplicación se utilizará GlassFish 4.1.1 y para las pruebas de carga se utilizará JMeter.
* **Resultados esperados:** Se espera que el sistema pueda atender cerca de 4000 solicitudes en menos de un minuto, además que en caso de un evento sísmico sea capaz de generar un boletín de emergencia en menos de 500 ms.
* **Duración y etapas:** Esta etapa finalizará para el 15 de Marzo de 2016. Deberá incluir pruebas de carga de la arquitectura planteada y la incorporación de la persistencia. Se realizará entregas semanales de avances progresivos. Inicialmente se llevará a cabo la incorporación de la persistencia.

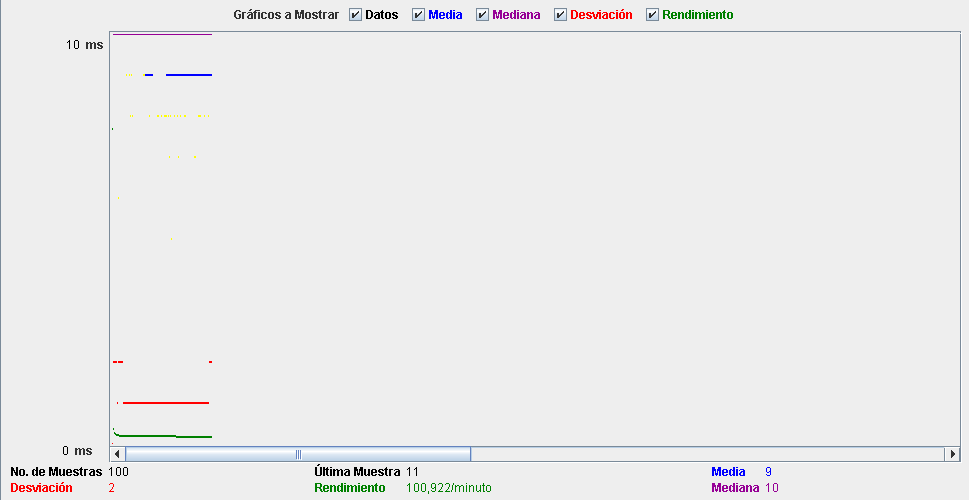
**Experimentación:**

Para este experimento se desarrolló y ejecutó las pruebas de carga de los siguientes requerimientos:

* Generación de boletín (Latencia menor a 500 milisegundos, probado con 100 Threads)
* La actualización de la información de un sensor en cambios de altura de una zona específica (Latencia de 1 segundo, probado con 100 Threds)
* La actualización de la información de 4000 sensores (Escalabilidad)

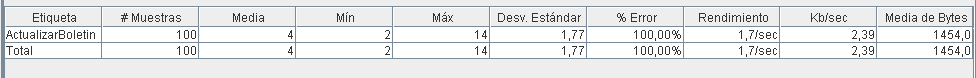
A continuación, se muestran algunos resultados obtenidos para estas pruebas y sus respectivos análisis:

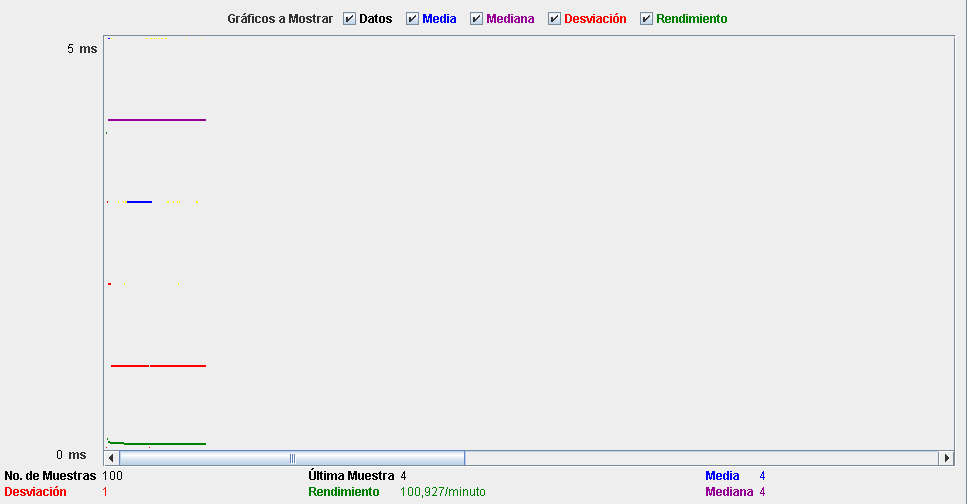




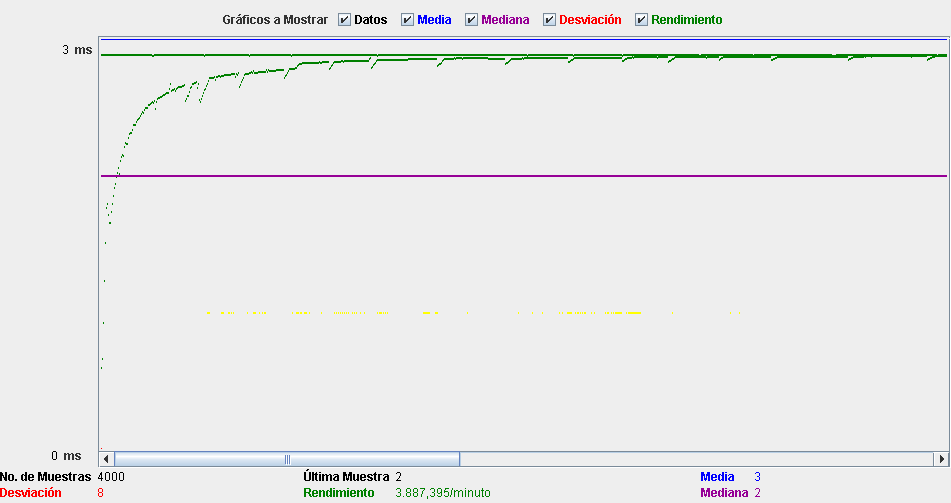
En la primera gráfica se puede apreciar que en la prueba de latencia para que generar un boletín, con 100 Threads, se obtiene una media de 9 ms de respuesta por cada ThreadGroup. Lo que garantiza que latencia en la generación de un boletín sea menor a 500 ms. Así mismo, la segunda gráfica lo constata.

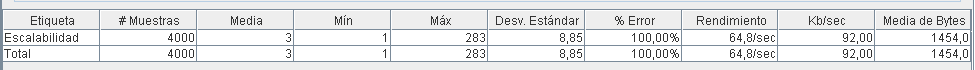
Ahora pasando a la segunda prueba de carga,





Tanto en la primera gráfica como en la segunda, de esta prueba, se puede constatar que hay una media de 4 ms. Es decir que el tiempo de latencia en la actualización de 100 sensores es de 4 ms por cada ThreadGroup. Cumpliendo el requerimiento no funcional establecido de la segunda prueba de carga. En la tercera prueba de carga, se obtuvo lo siguiente:





Se ve en la gráfica que tenemos un rendimiento de 3887,395 Threads por minuto. Lo que indica que no se cumple el requerimiento no funcional de escalabilidad. Ya que se debería lograr un total de 4000 threads por minuto.

**Post-experimentación:**

* **Resultados obtenidos:** Se obtuvieron resultados positivos para las dos pruebas de latencia. Sin embargo, para la prueba de escalabilidad no se cumplió con el requerimiento no funcional establecido.
* **Duración real:** A pesar que se cumplió con los tiempos establecidos, los resultados obtenidos no son del todo satisfactorios y la incorporación de las demás capas de la aplicación no se encuentra aún al 100%.
* **Artefactos construidos:** Se desarrollaron todos los artefactos mencionados en la pre-experimentación. Sin embargo, la incorporación de la persistencia no se ejecutó del todo, por un error en la actualización de tablas.
* **Análisis:** Se le atribuye los resultados negativos obtenidos en la prueba de escalabilidad al diseño de la arquitectura. En primer lugar, porque una petición tiene que viajar por las diferentes capas de la aplicación y además de esto una petición de inserción es más costosa que una de consulta. En segundo lugar, se les atribuye a los recursos computacionales con los que se cuenta.
* **Conclusiones:** Para que el tiempo de inserciones concurrentes en la aplicación disminuya es necesario acortar el camino por el que viaja el flujo de información de una petición.