TBC debe cumplir con dos escenarios de calidad primordiales, el primero sobre el rendimiento y el segundo sobre el desempeño. Se espera que es sistema sea capaz de responder a 4500 solicitudes en un tiempo no mayor a 5 segundos y se espera que el tiempo máximo de respuesta no supere 1 segundo. La primera prueba analizada fue el desempeño, esta prueba busca asegurar que el tiempo máximo de respuesta de la aplicación no exceda 1 segundo.

Para este experimento se cuento con un balanceador de carga, el cual re direccionaba peticiones a dos servidores (no pudieron ser más debido a limitaciones de la base de datos de Heroku) y estos se comunicaban con la base de datos. Se probaron todos los servicios que ofrece TBC y a continuación se presentan los resultados. Al igual que en el experimento anterior, se probaran primero los servicios de crear, luego las consultas get y por último los servicios de reporte y actualización.

La primera prueba de desempeño se realizó sobre los servicios de crear, como se ve en la *Tabla 1*, todos los tiempos medios cumplen con la expectativa de la prueba. Para el caso de crear tranvía, hubo un tiempo de 3,1 segundos, esto suponemos que fue la primera solicitud que se envió, una vez encontró al servidos las demás consultas se realizaron bajo condiciones ideales.

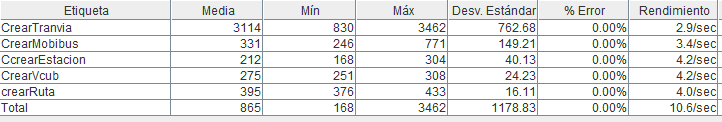


Tabla : Resultados servicios crear

Los resultados de la segunda prueba se muestran en la *Tabla 2*, en esta están registrados los resultados de las pruebas sobre los servicios get. En esta se evidencia el buen rendimiento de la aplicación, nuevamente se presentó un resultado superior al esperado en la primera prueba del grupo de hilos. En este caso, hay un 2% de tiempo sobre el esperado. Todas las demás pruebas se realizaron satisfactoriamente.

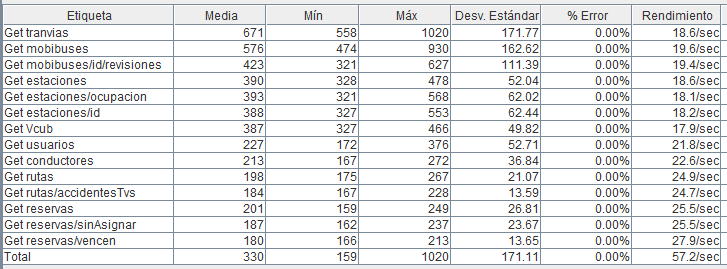


Tabla : Resultados servicios get

En cuanto a los servicios de actualización y reporte vimos un comportamiento anormal en el servicio que crea una nueva revisión a un tranvía. La media supera las expectativas del cliente, lo cual es un resultado poco satisfactorio. La desviación estándar de dicha prueba, como se puede ver en la tabla 3, es significativamente superior a las demás pruebas. Esto es un punto importante a tener en cuenta para el próximo experimento.

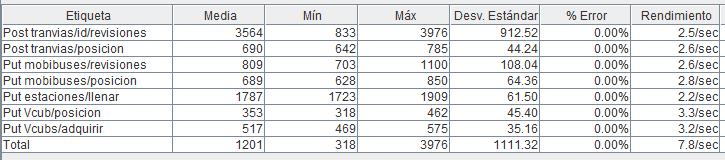


Tabla : Resultados servicios post/put

Todavía cabe señalar que en las tres pruebas de rendimiento, en las cuales se probaron todos y cada uno de los servicios, hubo 0 % de error. Esto quiere decir que bajo condiciones normales la aplicación está en capacidad de responder satisfactoriamente a todas las solicitudes de los usuarios.

La prueba de escalabilidad se ejecutó en un ambiente de sobrecarga, en la cual se enviaron 5000 solicitudes casi de inmediato para poder analizar el comportamiento de respuesta de la aplicación. En la Ilustración 1 (Transacciones por segundo Vs Tiempo), se ve que hubo una ruptura en el comportamiento donde una considerable parte de las consultar resulto en falla. Se asume que es debido a limitaciones que impone Heroku por tratarse de una versión gratuita, donde deja de recibir solicitudes después de cierta cantidad de transacciones por segundo.

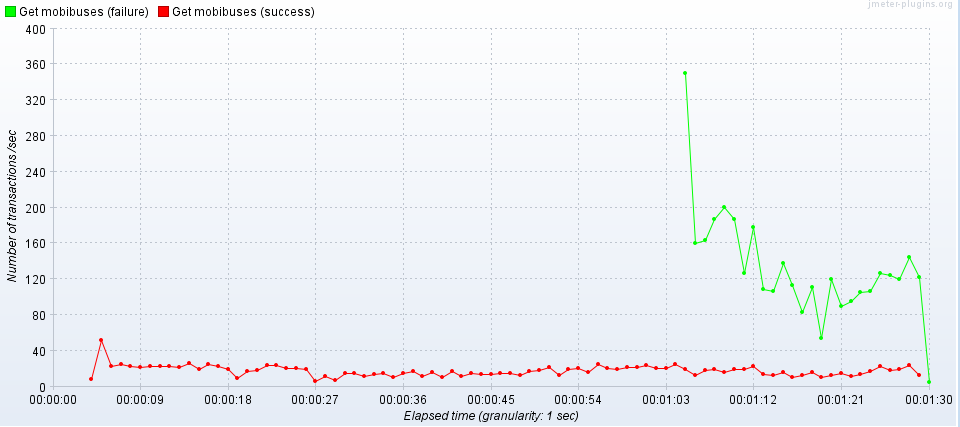


Ilustración : Esc1. # Transacciones/segundo Vs Tiempo

Adicional a lo anterior, podemos ver en la Ilustración 2 (Tiempo de respuesta Vs. Tiempo), que en el mismo instante en que empezaron a ocurrir fallas se redujo notablemente el tiempo de respuesta. Esto pudo ser por que la solicitud viajo hasta Heroku, en donde está alojada la base de datos, y esta respondió con error por lo que vemos un comportamiento relativamente similar después de esta ruptura.

Ahora bien, analizando con más detalle esta gráfica, se ve como los tiempos de respuesta en comienzo eran algo cercanos unos de otros, pero a medida paso el tiempo se empezaron a alterar resultados altos y bajos. Esta diferencia en gran medida ha de deberse a la de respuesta de un nodo servidor con otro. Dicha diferencia que se presenta en los tiempos pareciera ser originado debido a que un servidos empezó a crear cola de solicitudes y por ende dejo de atenderlas de inmediato, de esta manera cada vez el tiempo de respuesta era mayor hasta el momento en que Heroku dejo de atender gran parte de las solicitudes.

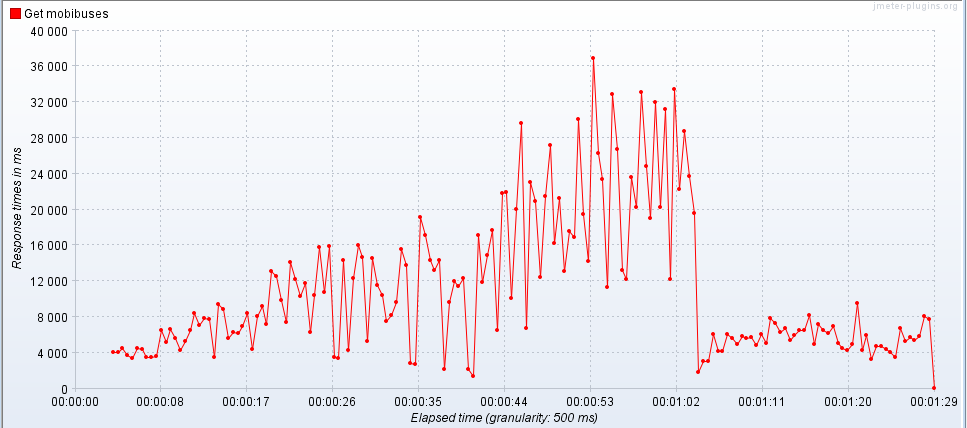


Ilustración : Esc2. Tiempo de respuesta Vs. Tiempo