**PRUEBA RENDIMIENTO HU-03**

En este apartado solo se va a probar un escenario ya que el de borrar el libro correctamente se prueba junto a la HU-01 de forma que tras añadir un libro se borra para evitar problemas que hemos encontrado durante el desarrollo de las pruebas.

**Stress test:** el mínimo número de usuarios concurrentes que no pueden ser soportados en los escenarios es de aproximadamente 10000 usuarios “en rampa” durante 10 segundos (téngase en cuenta que es un solo escenario a diferencia del resto de análisis que son 2)

El motivo del mal rendimiento es, como se comenta en análisis de historias de usuario posteriores el consumo de una API que ofrece un mal rendimiento.

Por dicha razón la interfaz de red actúa como cuello de botella.

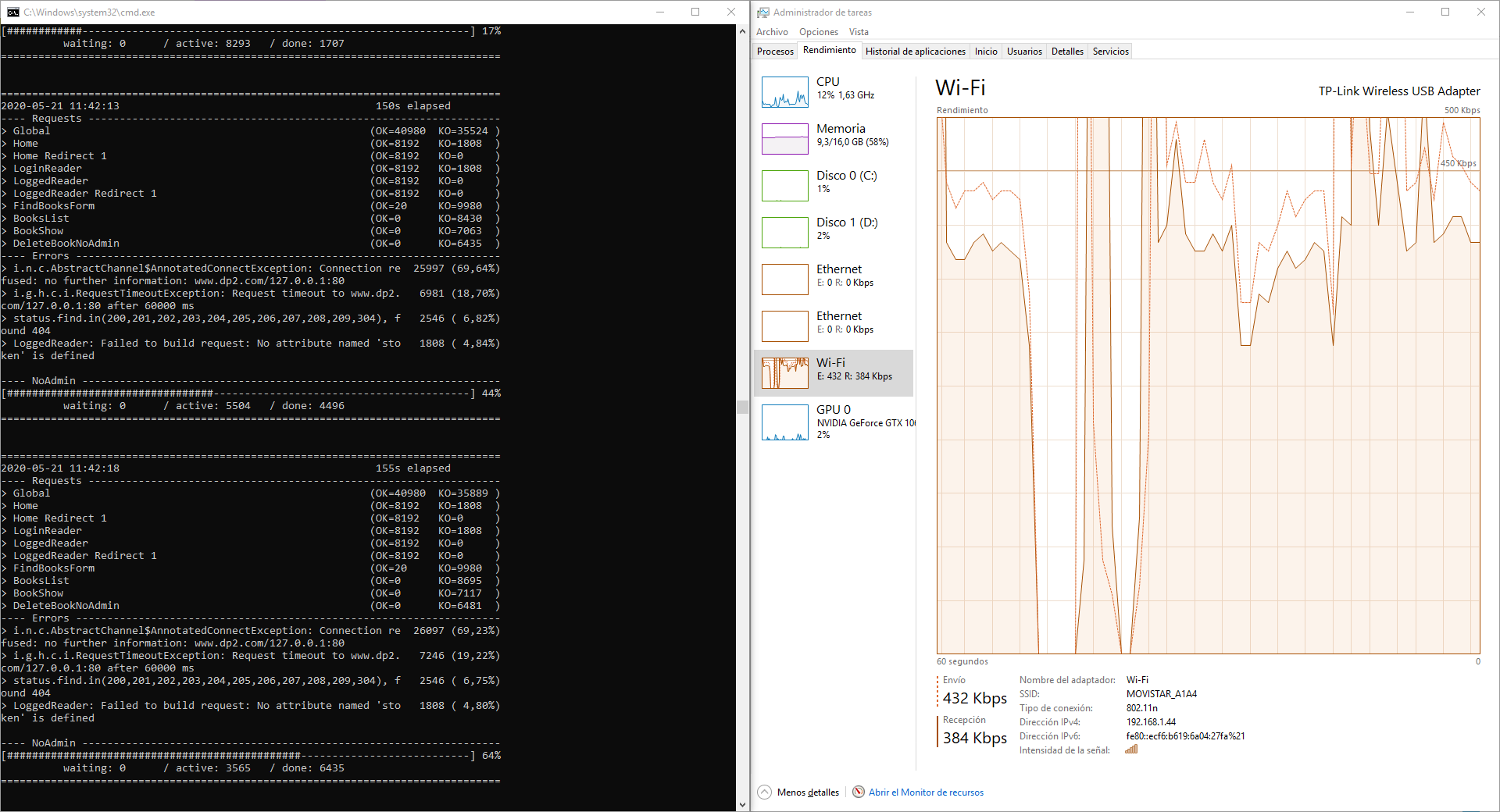


Ilustración : cuello de botella HU-03

**Load test:** debido al mal rendimiento de la API que tiene un tiempo de respuesta muy elevado y además solo acepta 20 peticiones y después se satura, el máximo número de usuarios que soporta este escenario con un rendimiento óptimo es de 71

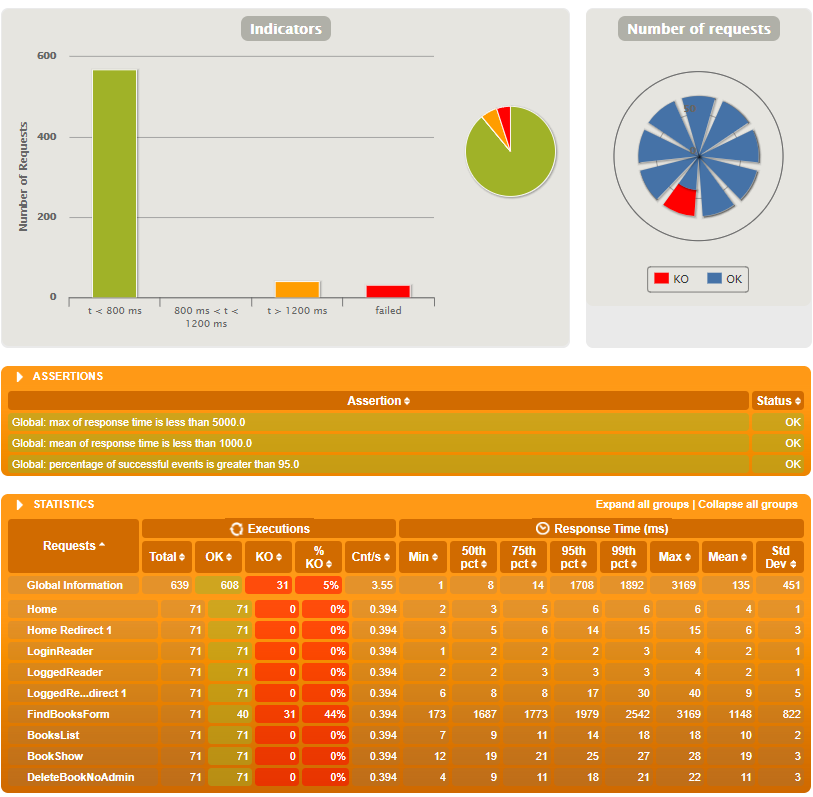


Ilustración : Test de carga HU-03

**PRUEBA RENDIMIENTO HU-06**

**Test de stress:** el mínimo número de usuarios concurrentes que no pueden ser soportados en los escenarios está alrededor de los 15000 usuarios en cada uno de los dos escenarios, repartidos en 10 segundos.

Al comenzar la ejecución se desborda la CPU, que es el cuello de botella, y a medida que avanzan las peticiones se va estabilizando.

Si se disminuyen los usuarios concurrentes la aplicación ya sería capaz de ir empezando a soportar los escenarios.

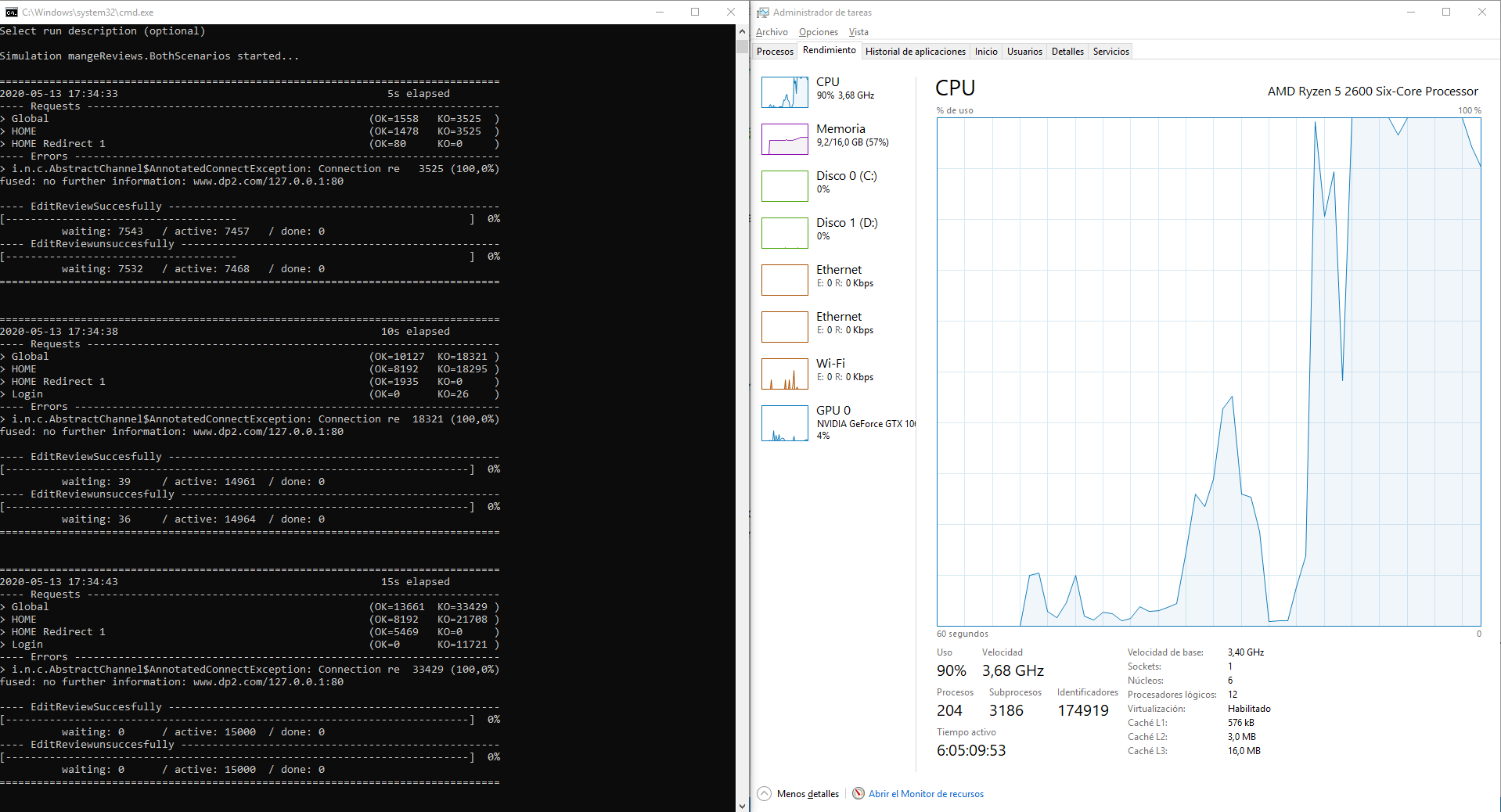


Ilustración 3:cuello de botella HU-06

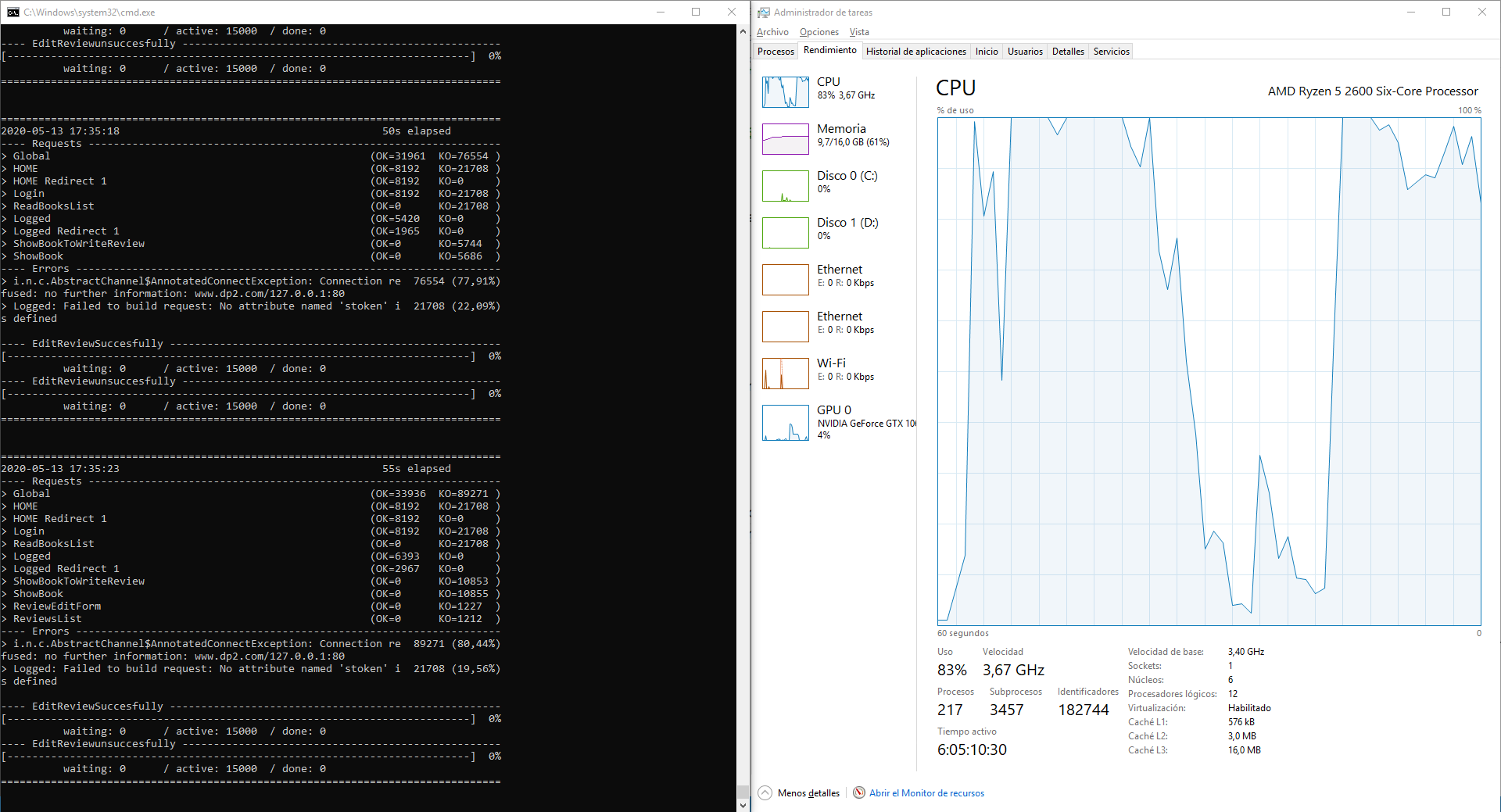


Ilustración 4: cuello de botella HU-06

**Load test:** el máximo número de usuarios al que se puede ofrecer servicio con un buen rendimiento (95% de peticiones OK, media 1s la respuesta y 5s máximo tiempo respuesta) es de **3000 usuarios repartidos durante 100 segundos**.



Ilustración 5: test de carga HU-06

**PRUEBA RENDIMIENTO HU-08**

Los resultados obtenidos en estos escenarios son peores porque se entra en la página de buscar libros en la que se llama a una API externa que devuelve un poema aleatorio.

**Stress test:** el mínimo número de usuarios concurrentes que no pueden ser soportados en los escenarios ronda alrededor de los 5000 usuarios en cada uno de los dos escenarios, repartidos en 10 segundos.

Al comenzar la ejecución cuando se accede a la página en la que se realiza la llamada a la API externa la interfaz de red WIFI recibe una gran carga de trabajo y creemos que puede ser el cuello de botella.

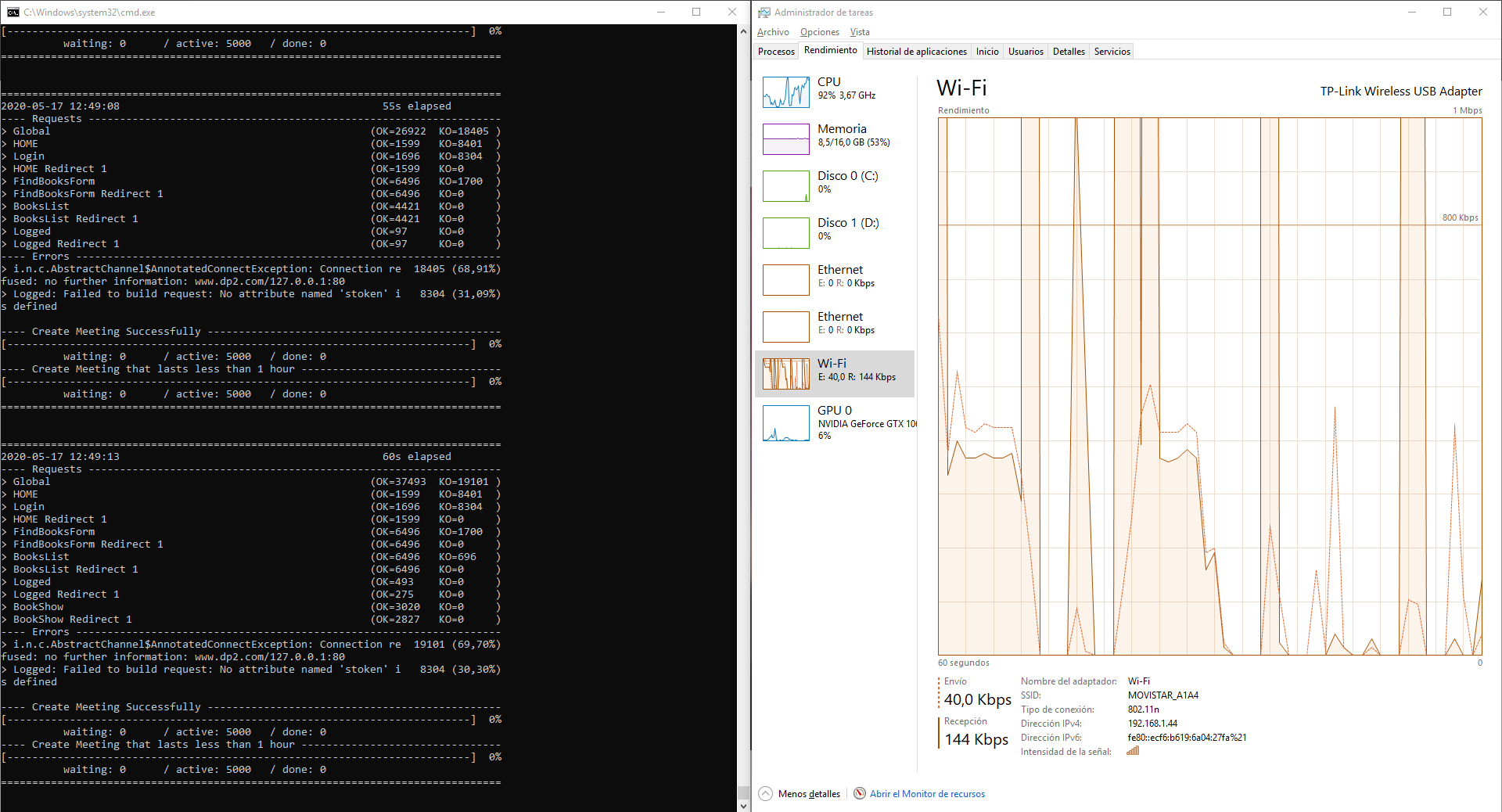


Ilustración 6: cuello de botella HU-08

Cabe destacar que se ha usado el feeder CreateMeetingStressFeeder.csv para los datos de las reuniones que se pretenden crear correctamente.

**Load test:** debido al uso de la API solo se pude dar un rendimiento óptimo a 33 usuarios repartidos durante 10 segundos. A partir de ahí los tiempos de respuesta medio y máximo, así como el porcentaje de respuestas fallidas se dispara, ya que la API (<https://poemist.github.io/poemist-apidoc/>) se satura y empieza a devolver fallos.

Si nos fijamos durante la ejecución las primeras 20 peticiones a la API funcionan con normalidad, aunque lentas, pero después deja de aceptar las peticiones. Suponemos que tienen “capado” el tráfico a un usuario y nos plantearemos cambiar la API por otra que ofrezca un rendimiento aceptable, ya que el resto de la aplicación aguanta una cara muchísimo mayor.

La petición que contiene la llamada a la API, como se puede deducir por su mal rendimiento, es FindBooksForm.

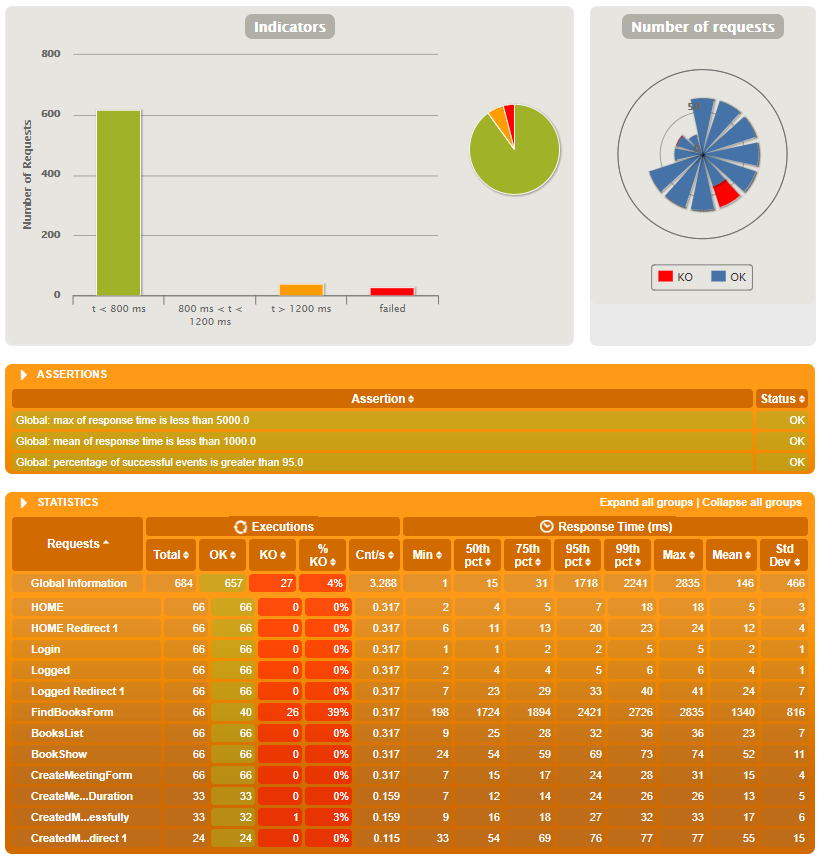


Ilustración 7: test de carga HU-08

Se ha usado un feeder para dar los datos para crear la reunión: CreateMeetingLoadFeeer.csv

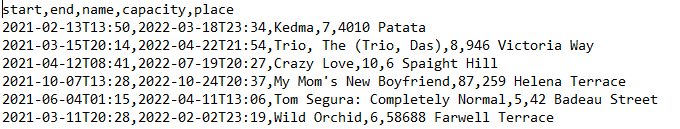


Ilustración : ejemplo feeder

Y se pueden ver los resultados en la BD que concuerdan con los datos del feeder:



Ilustración : resultado en BD

**PRUEBA RENDIMIENTO HU-11**

**Stress test:** el mínimo número de usuarios concurrentes que no pueden ser soportados en los escenarios ronda alrededor de los 10000 usuarios en cada uno de los dos escenarios, repartidos en 10 segundos. Concuerda con los resultados de la mayoría de las pruebas realizadas (en mi máquina).

Al comenzar la ejecución se desborda la CPU, que es el cuello de botella, pero pronto se recupera. Por ello, en cuanto se bajen un poco los usuarios la aplicación ya será capaz responder, aunque no con un rendimiento óptimo todavía.

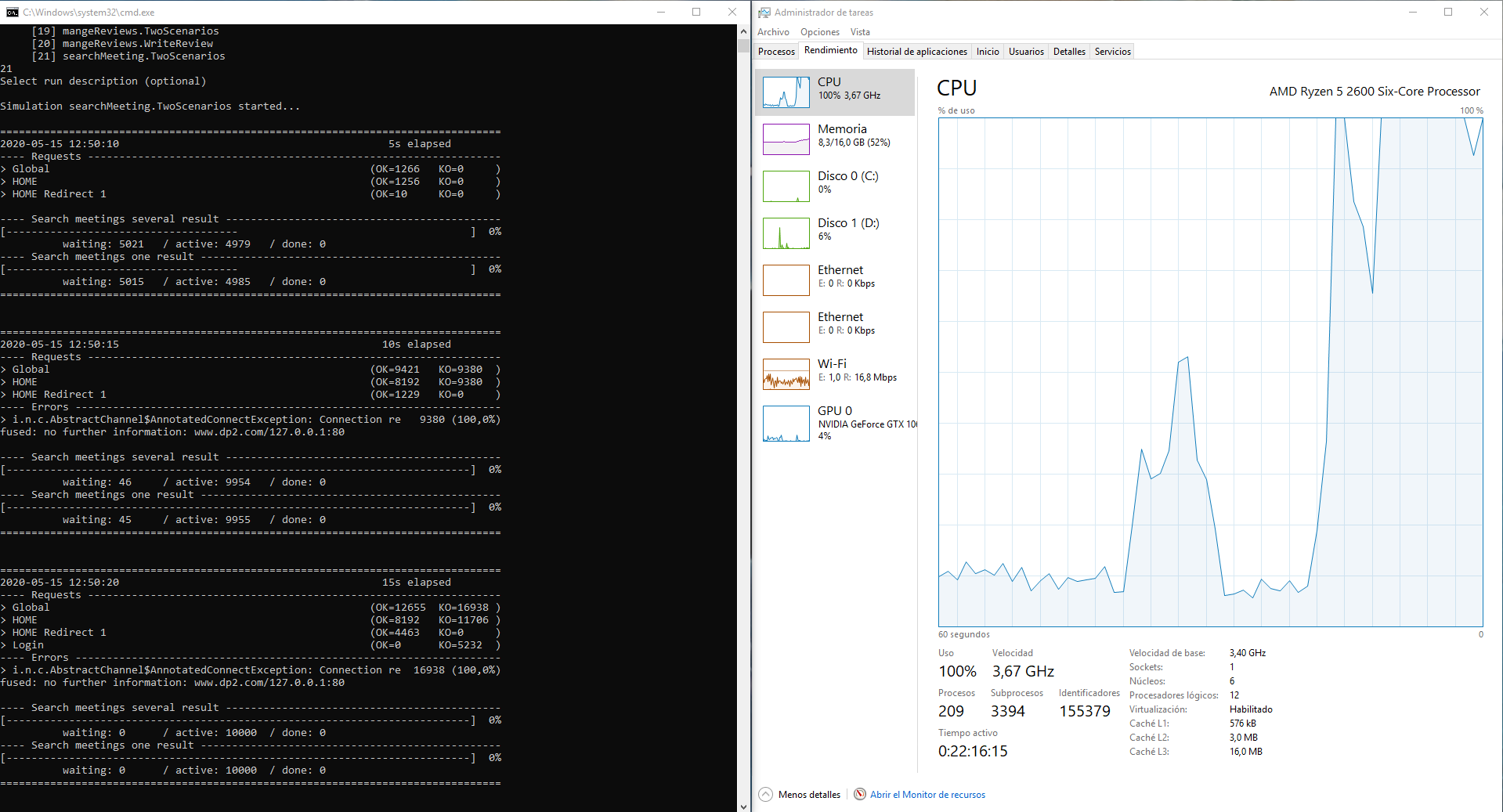


Ilustración 10: cuello de botella HU-11

**Load test:** el máximo número de usuarios al que se le puede ofrecer un rendimiento aceptable es de 3500 usuarios en cada escenario repartidos en 100 segundos.

Cabe destacar que en la petición MeetingsSearchForm se realiza una llamada a una API externa (<https://github.com/lukePeavey/quotable>). Por tanto, dicha API ofrece un servicio con una calidad mucho mayor que la citada en el análisis de la HU-08, que soportaba como máximo solo 90 usuarios en lugar de 3500.

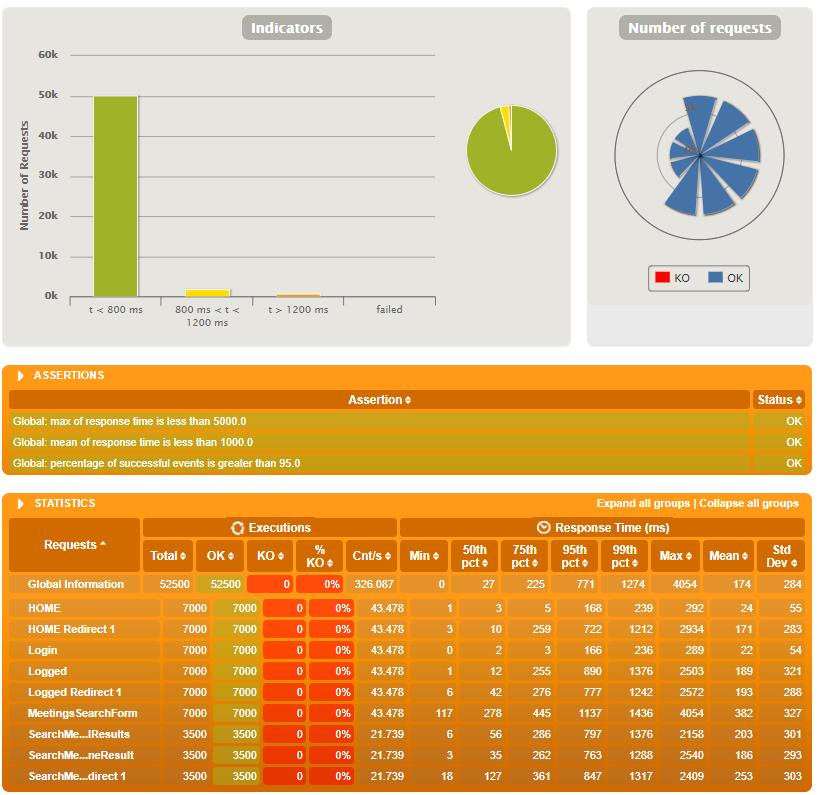


Ilustración 11: test de carga HU-11

**PRUEBA RENDIMIENTO HU-22**

Ambos escenarios de esta historia de usuario dependen de los datos de una API externa (<https://api.itbook.store/>) por lo que cabe esperar que el rendimiento sea más bajo.

**Stress test:** el mínimo número de usuarios concurrentes que no pueden ser soportados en los escenarios ronda alrededor de los 7000 usuarios en cada uno de los dos escenarios, repartidos en 10 segundos.

Al comenzar la ejecución se desborda la CPU, que es el cuello de botella, pero pronto se recupera. En cuanto bajan los usuarios (por ejemplo a 5000) el rendimiento ya empieza a ser aceptable, aunque sigue sin ser óptimo.

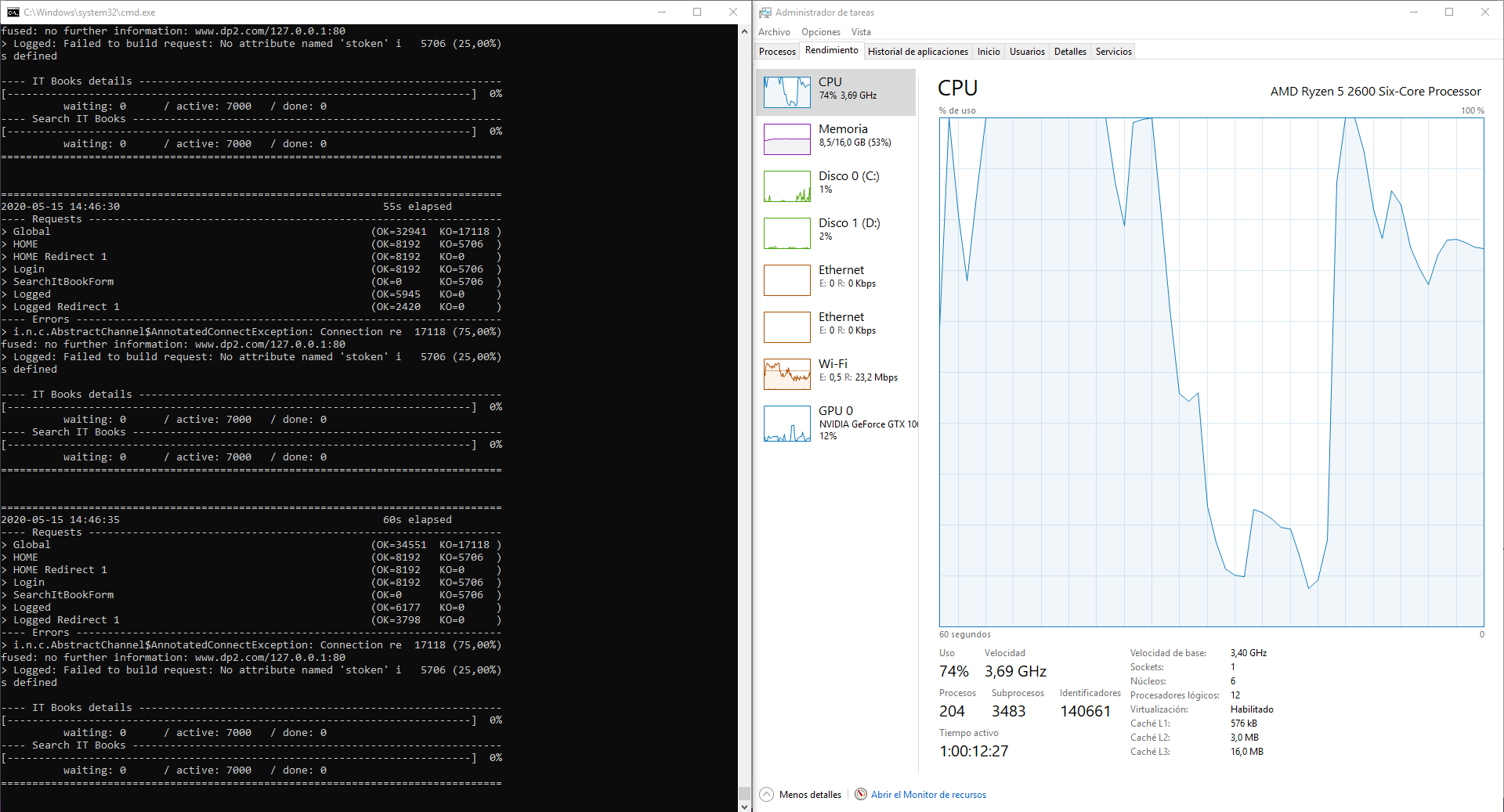


Ilustración 12: cuello de botella HU-22

**Load test:** el máximo número de usuarios al que se le puede ofrecer un rendimiento aceptable es de 1000 usuarios en cada escenario repartidos en 100 segundos.

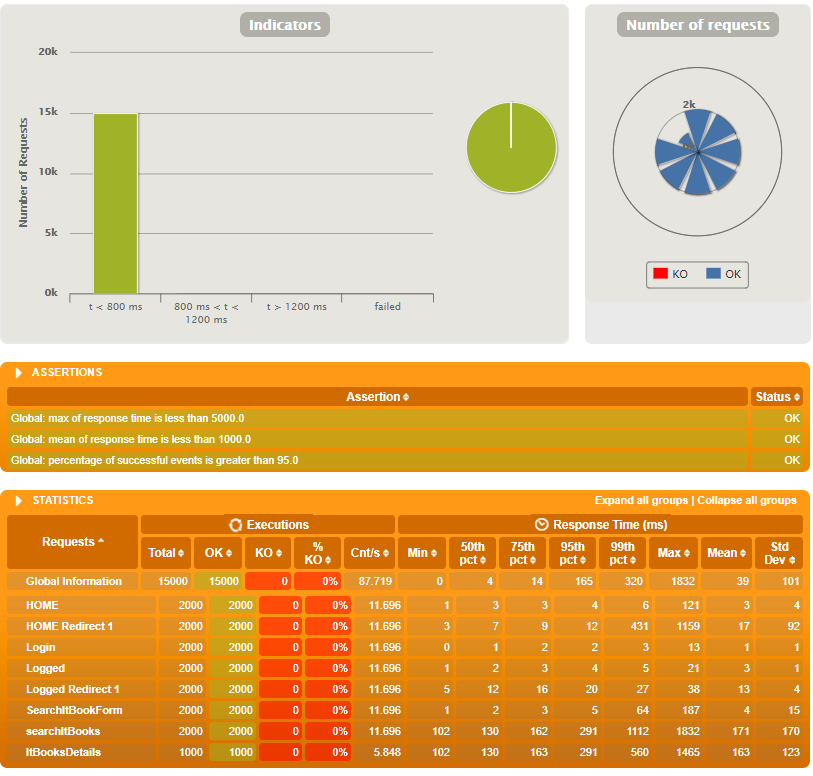


Ilustración 13: test de carga HU-22

En cuanto se suben los usuarios a 1500 el tiempo de respuesta máximo se dispara en torno a los 9 segundos.