|  |  |
| --- | --- |
| PET-CLINIC (Rendimiento) | GRUPO 13  Jesús Andrés Rico Catalán  Carlos Albalat Heredia |

Índice

[Introducción 2](#_Toc42187087)

[Rendimiento del sistema 3](#_Toc42187088)

[HU-1,3,5: Diagnosis 3](#_Toc42187089)

[HU2: Diagnosis 5](#_Toc42187090)

[HU-4,14: Diagnosis 9](#_Toc42187091)

[HU6: Diagnosis 11](#_Toc42187092)

[HU7: Diagnosis 14](#_Toc42187093)

[HU8: Diagnosis 17](#_Toc42187094)

[HU9: Diagnosis 19](#_Toc42187095)

[HU10: Diagnosis 20](#_Toc42187096)

[HU11: Diagnosis 20](#_Toc42187097)

[HU12: Diagnosis 21](#_Toc42187098)

[HU13: Diagnosis 21](#_Toc42187099)

[HU6 vs HU8: Diagnosis 21](#_Toc42187100)

[Conclusión 25](#_Toc42187101)

# Introducción

Para comenzar se va a dejar constancia de los equipos utilizados para realizar las pruebas de rendimiento de la aplicación:





Es importante mencionar que dependiendo de las especificaciones del equipo en donde se corran las pruebas de rendimiento pueden variar considerablemente los resultados, pero independientemente de los usuarios que soporte nuestro equipo se puede diagnosticar los cuellos de botellas que puede producir el sistema, así como analizar la eficiencia de nuestras HUs.

Para comprobar el rendimiento del sistema se han realizado bajo las “assertions” siguientes:

 setUp(

  transportarNeg.inject(rampUsers(280) during (100 seconds)),

  transportarPos.inject(rampUsers(280) during (100 seconds))

  ).protocols(httpProtocol)

    .assertions(

        global.responseTime.max.lt(5000),

        global.responseTime.mean.lt(1000),

        global.successfulRequests.percent.gt(95)

     )

Consideramos que nuestro sistema tiene un rendimiento adecuado cuando se satisfacen el 95% de las peticiones y la media de los tiempos de respuestas de dichas peticiones no superan el segundo. El tiempo de respuesta máximo verifica que en 5 segundos todas las respuestas han sido satisfechas, cuando no se cumple dicha condición podemos analizar de manera sencilla donde están los cuellos de botella así como TimeOut cuando el sistema es sometido a mucho estrés. Dependiendo del dispositivo en el que se ejecuta y la naturaleza de la prueba el rendimiento máximo de cada historia será diferente.

# Rendimiento del sistema

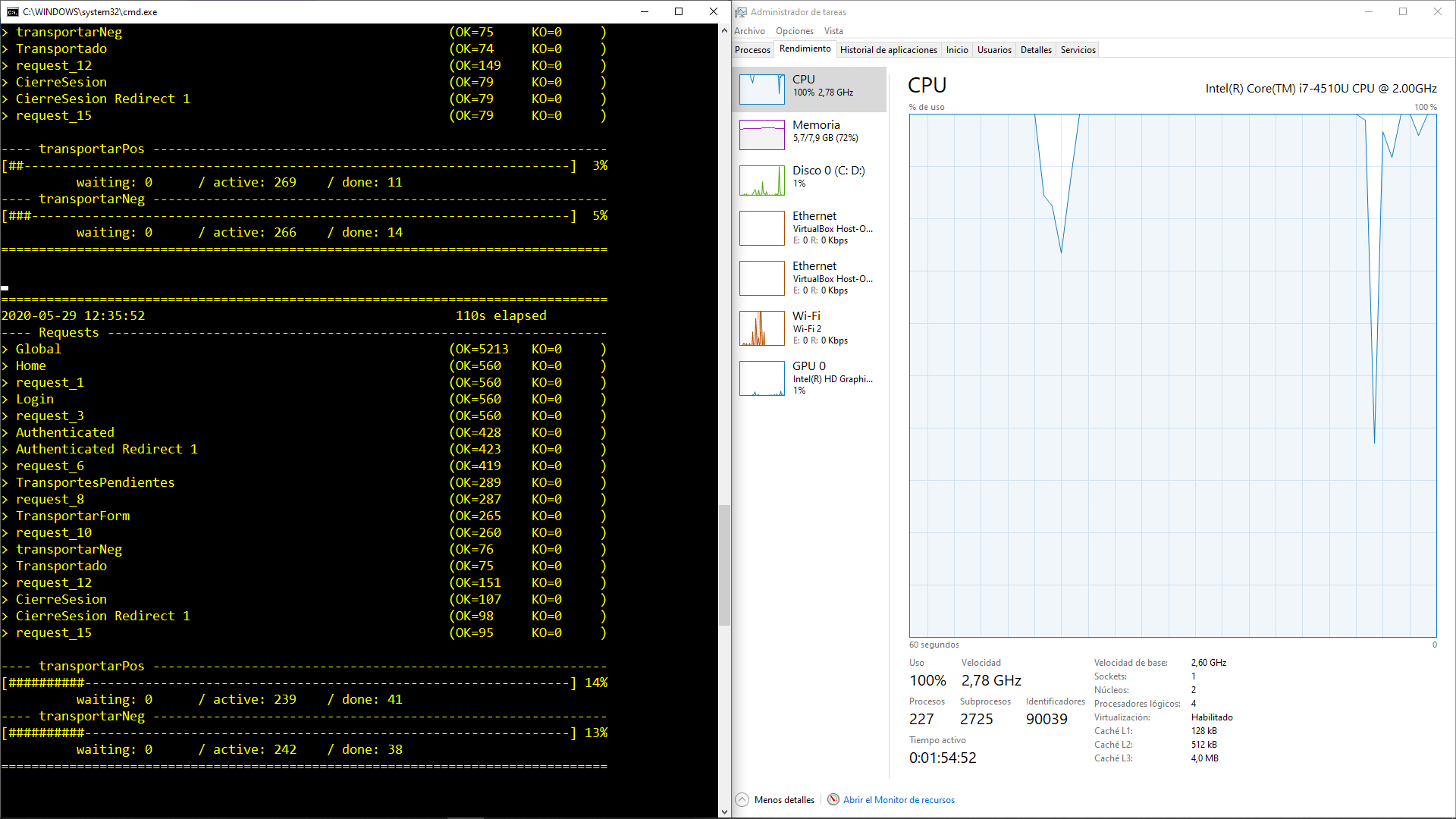
Algunos test de rendimiento incluyen varias HUs en el mismo test debido a la naturaleza de esta. Debido a que cada compañero posee un equipo con distintas especificaciones varía considerablemente el rendimiento de nuestra aplicación.

## HU-1,3,5: Diagnosis

El siguiente test engloba estas HUs:

* El secretario debe poder transportar animales consultando en la base de datos animales por transportar.
* El secretario debe poder listar de manera sencilla los animales que necesitan un transporte.
* El secretario solo puede solicitar un transporte con varios animales sí esos animales pertenecen al mismo dueño.

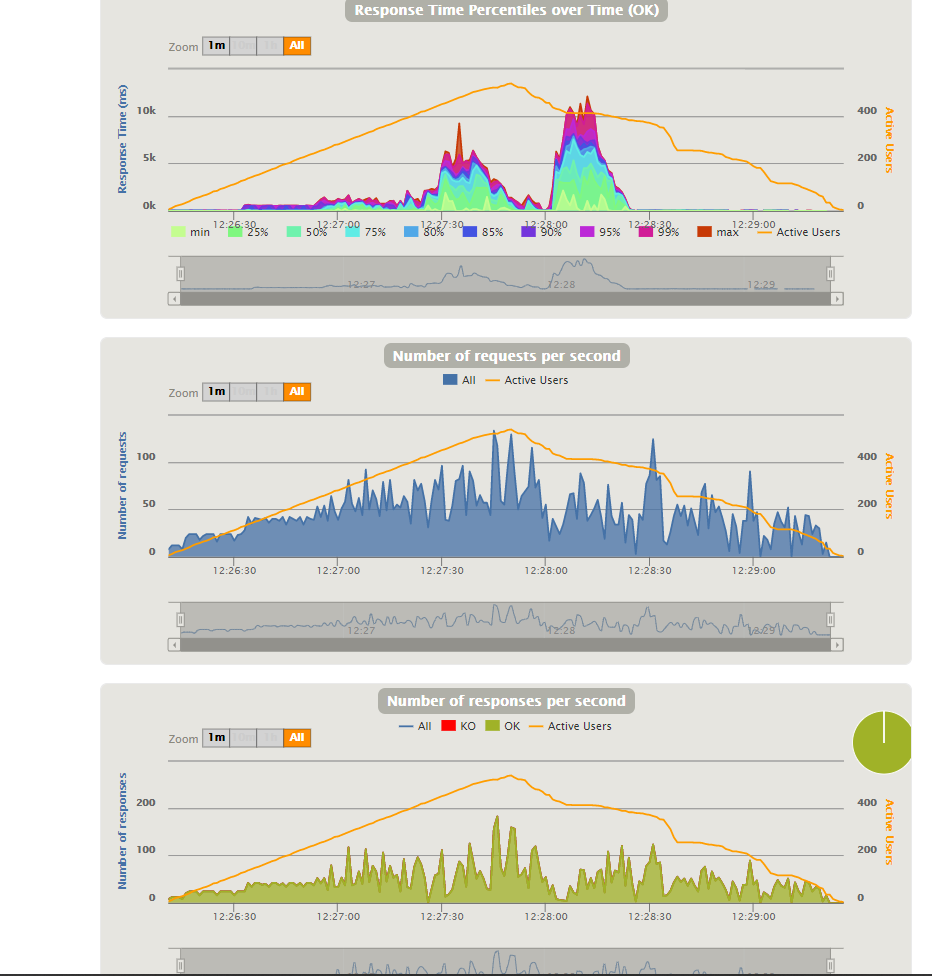
El máximo número de usuarios concurrentes que soporta de manera correcta es 560. Un aumento de 20 usuarios concurrente provoca una gran saturación provocando que el sistema para estas historias sea inviable usar la aplicación. La causa de que nuestro servidor deje de responder de manera óptima es debido a que la CPU no es capaz de procesar tantas peticiones concurrentemente propiciando cuellos de botellas y por lo tanto muchos TimeOuts como se puede ver en la siguiente captura:



\*En este test se enfrenta un escenario positivo de transportar y otro negativo.

Cabe destacar que una vez se mejore las especificaciones de la CPU aumentará considerablemente el número de usuario concurrentes del sistema llegando a un punto en el que habrá que mejorar la RAM o aumentar su capacidad ya que prácticamente llega al 80% de su funcionamiento.





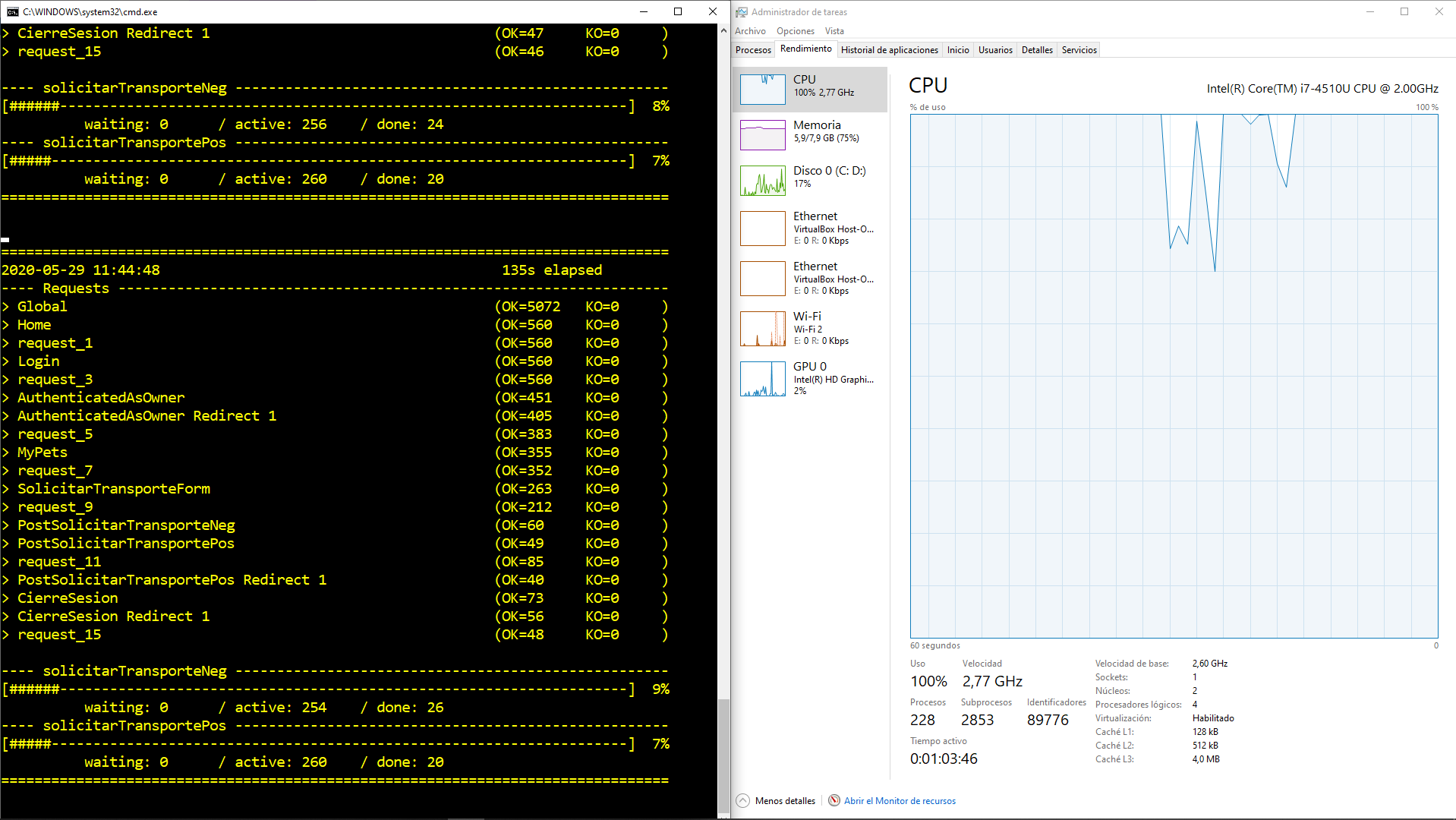
En el report generado por Gatling podemos comprobar que el sistema satisface como mínimo el 95% de peticiones http que genera el test. También se verifica que la media de las peticiones es menor a un segundo aunque se pueden observar que al someter a cierto estrés hay una serie de peticiones que necesitan consultar la base de datos y por lo tanto un mayor procesamiento generando cuellos de botellas como son las autenticación y los post (edición) de la entidad transporte pero son casos puntuales.

## HU2: Diagnosis

El siguiente test engloba esta HU:

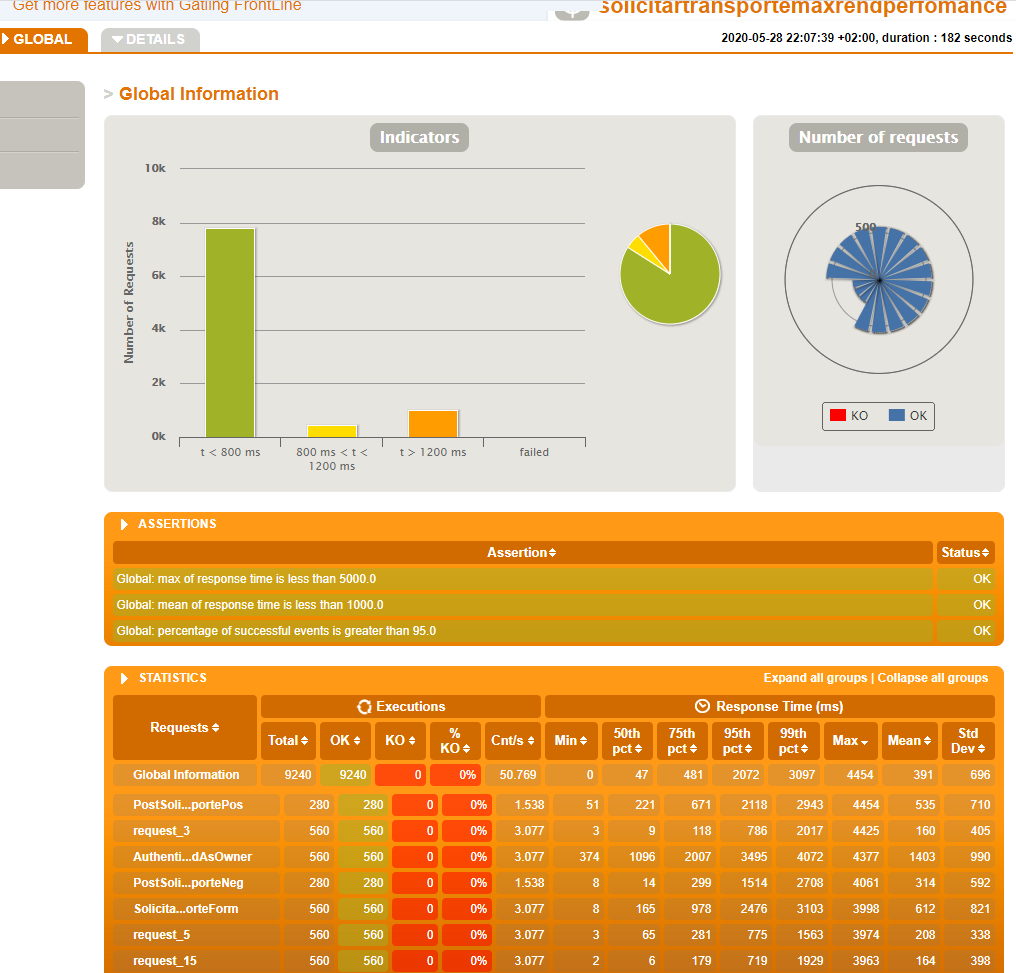
* El propietario puede solicitar transportar a su animal.

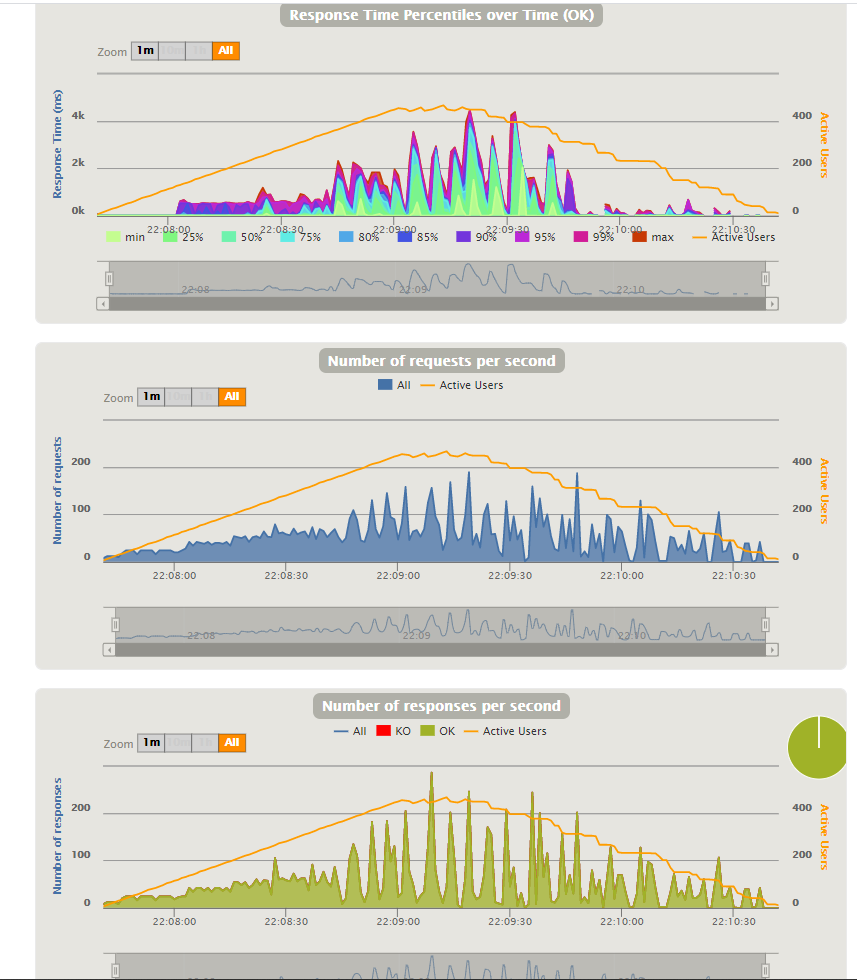
El máximo número de usuarios concurrentes que se soporta es de 560. Un aumento de 35 usuarios concurrente provoca una gran saturación provocando que el sistema para esta historia sea inviable usar la aplicación. La causa de que nuestro servidor deje de responder de manera óptima es debido a que la CPU no es capaz de procesar tantas peticiones concurrentemente propiciando cuellos de botellas y por lo tanto muchos TimeOuts como se puede ver en la siguiente captura:

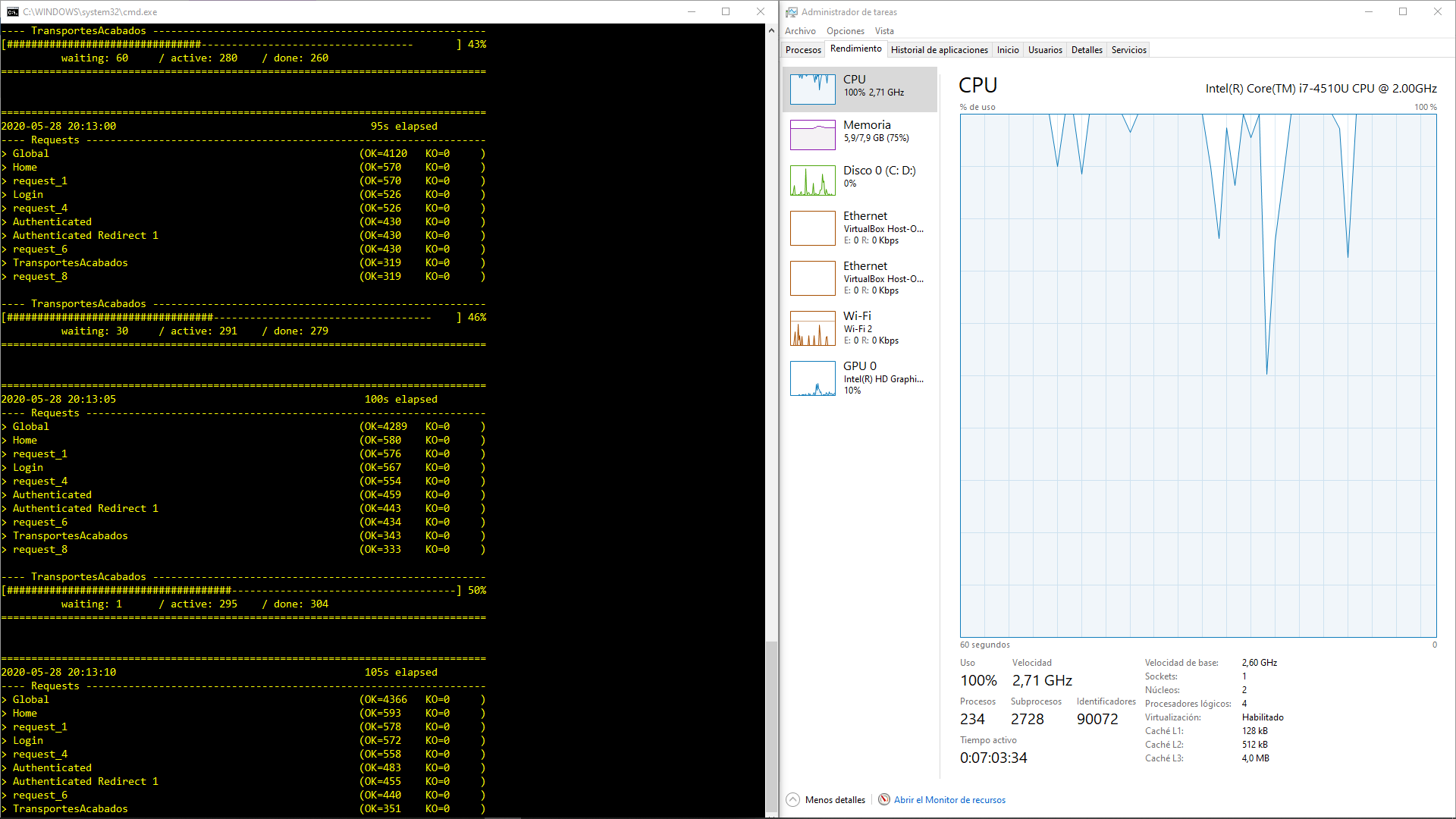


\*En este test se enfrenta un escenario positivo de solicitarTransporte y otro negativo.

Cabe destacar que una vez se mejore las especificaciones de la CPU aumentará considerablemente el número de usuario concurrentes del sistema llegando a un punto en el que habrá que mejorar la RAM o aumentar su capacidad ya que prácticamente llega al 80% de su funcionamiento.





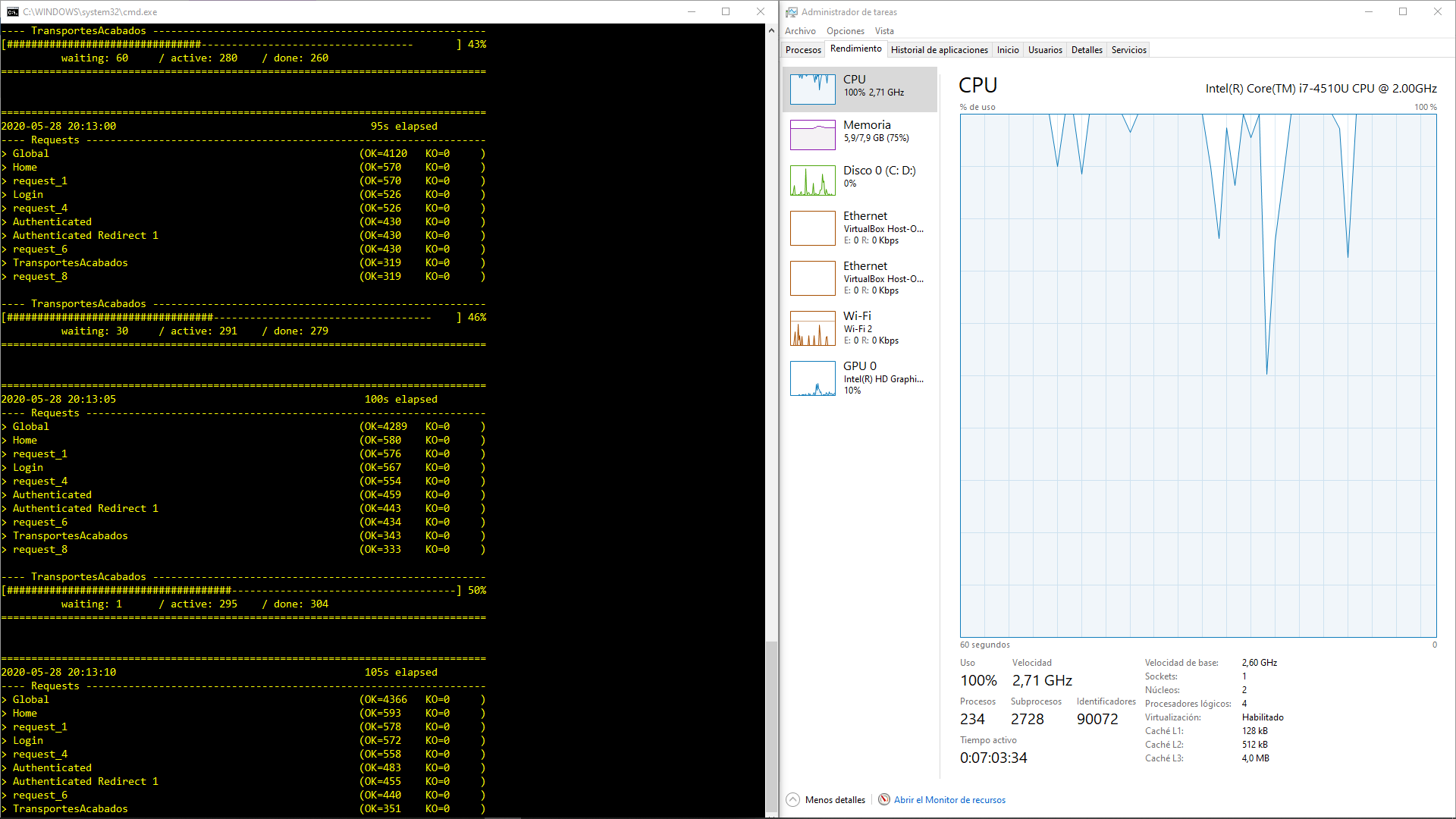
En el report generado por Gatling podemos comprobar que el sistema satisface como mínimo el 95% de peticiones http que genera el test. También se verifica que la media de las peticiones es menor a un segundo y que ninguna petición supera un tiempo de respuesta mayor a 5 segundos. Por lo tanto, se podría decir que el comportamiento es idóneo ya que apenas se producen cuellos de botella.

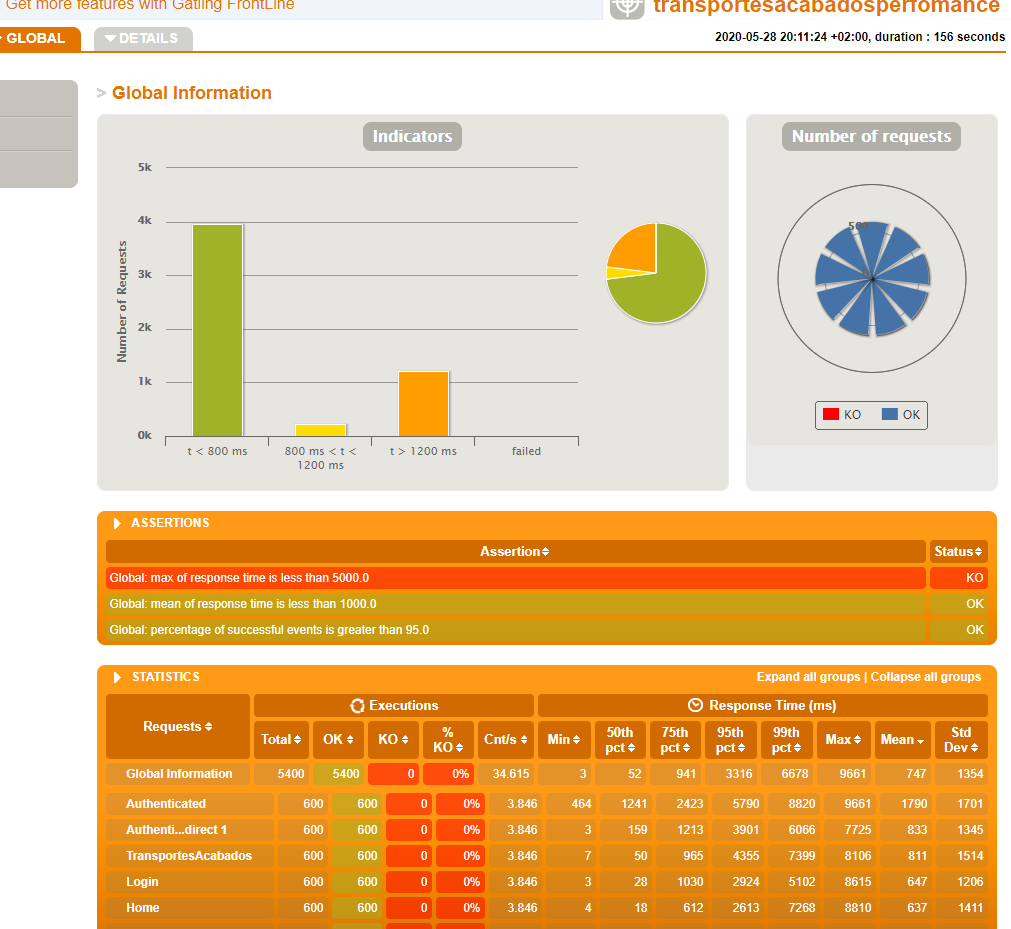
## HU-4,14: Diagnosis

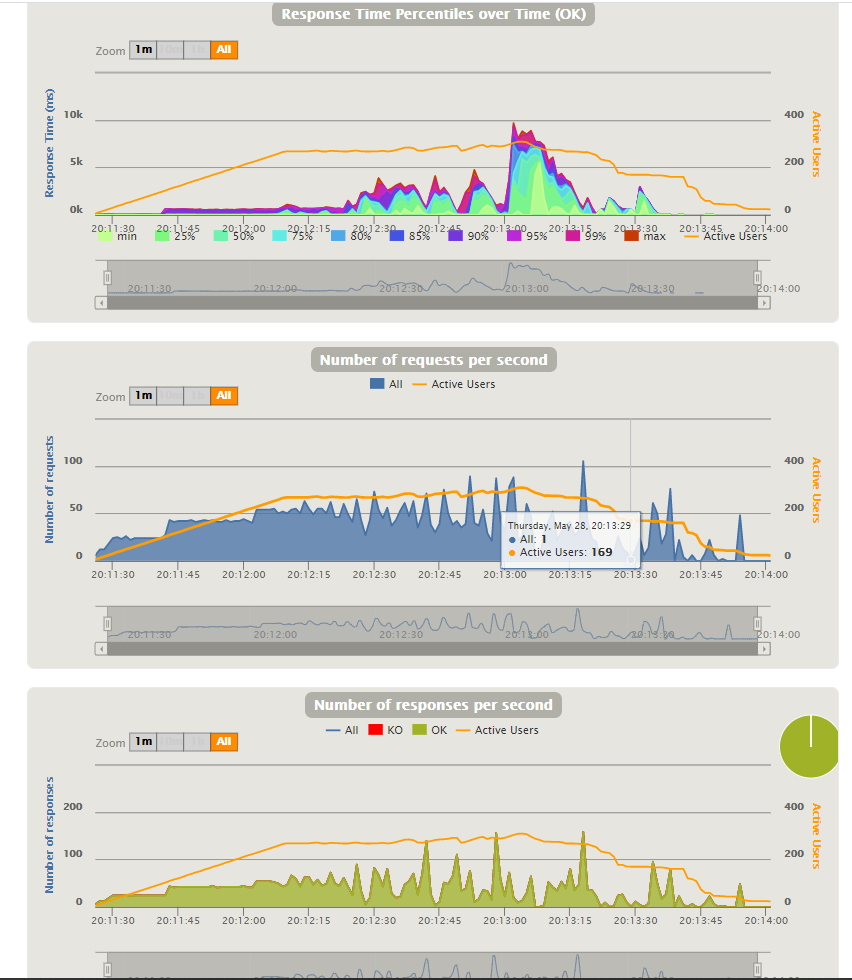
El siguiente test engloba esta HU:

* El propietario debe poder ver todos sus transportes acabados.
* El secretario debe poder ver los animales que han sido transportados.

Debido a la naturaleza de la HU que solo consulta la DB y no realiza ningún tipo de inserción o eliminación, el máximo número de usuarios concurrentes es algo mayor a las otras historias de transportes, esta es de 600 usuarios. Un aumento de 30 usuarios concurrente provoca una gran saturación provocando que el sistema para esta historia sea inviable usar la aplicación. La causa de que nuestro servidor deje de responder de manera óptima es debido a que la CPU no es capaz de procesar tantas peticiones concurrentemente propiciando cuellos de botellas y por lo tanto muchos TimeOuts, como se puede ver en la siguiente captura:







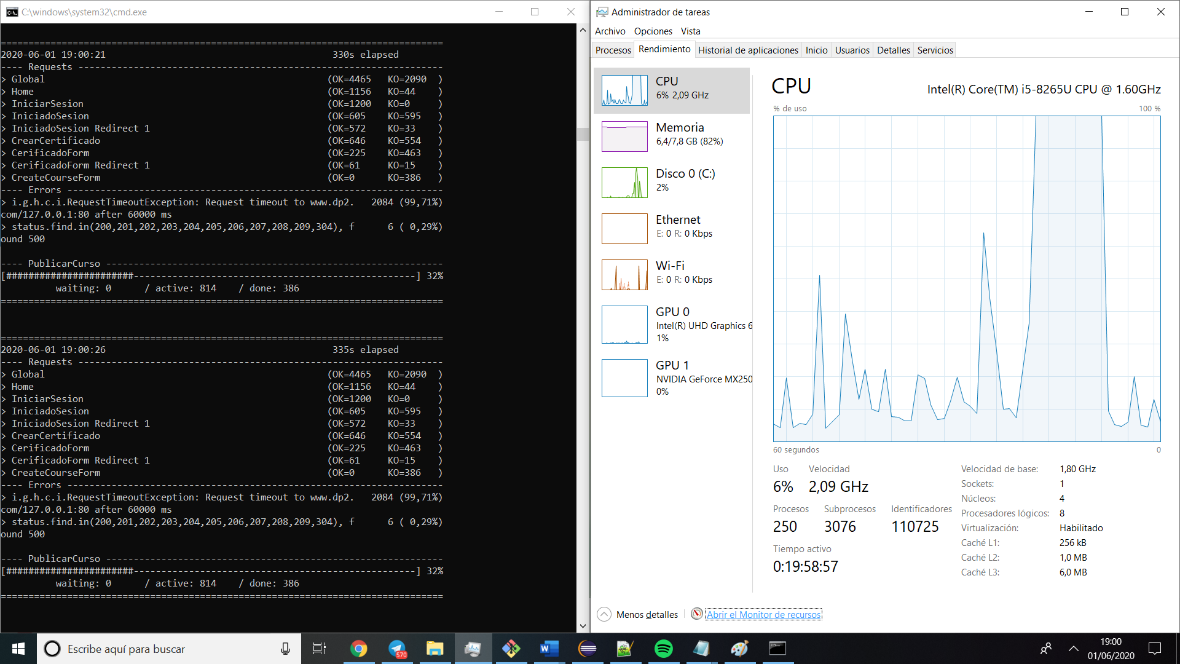
En el report generado por Gatling podemos comprobar que el sistema satisface como mínimo el 95% de peticiones http que genera el test. También se verifica que la media de las peticiones es menor a un segundo aunque se pueden observar que al someter a cierto estrés hay una serie de peticiones que necesitan consultar la base de datos y por lo tanto un mayor procesamiento generando cuellos de botellas como son las autenticación y la query de transportes acabados pero son casos puntuales.

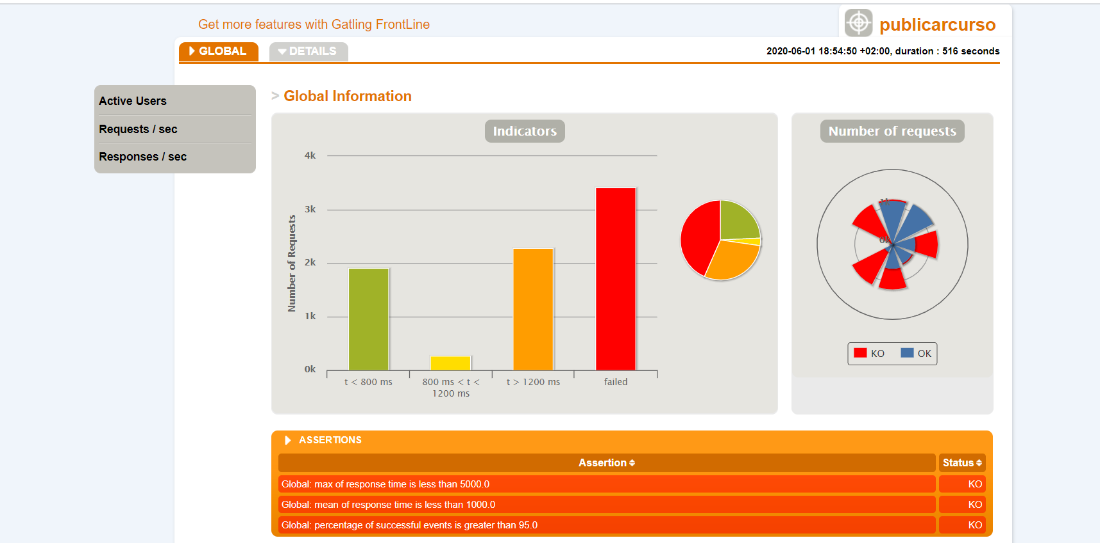
## HU6: Diagnosis

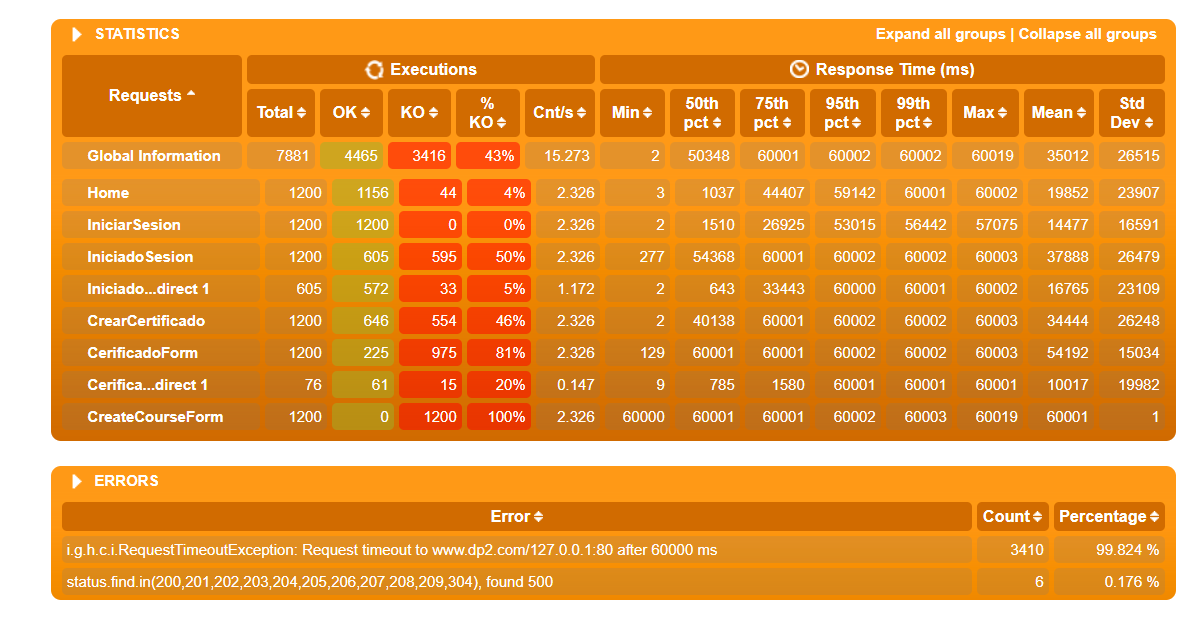
Este test de rendimiento se centra en la siguiente HU:

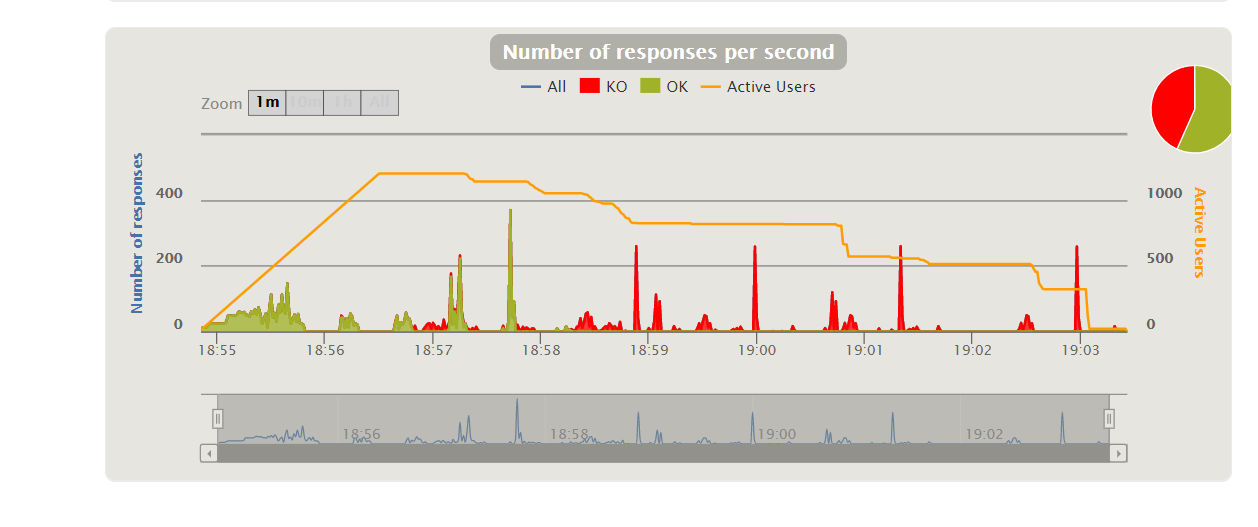
* El secretario debe poder publicar cursos.

El primer parámetro a identificar es el mínimo de usuarios concurrentes que no puede soportar esta HU junto a una captura de pantalla sobre el rendimiento del sistema mostrando el cuello de botella en la CPU y/o RAM.



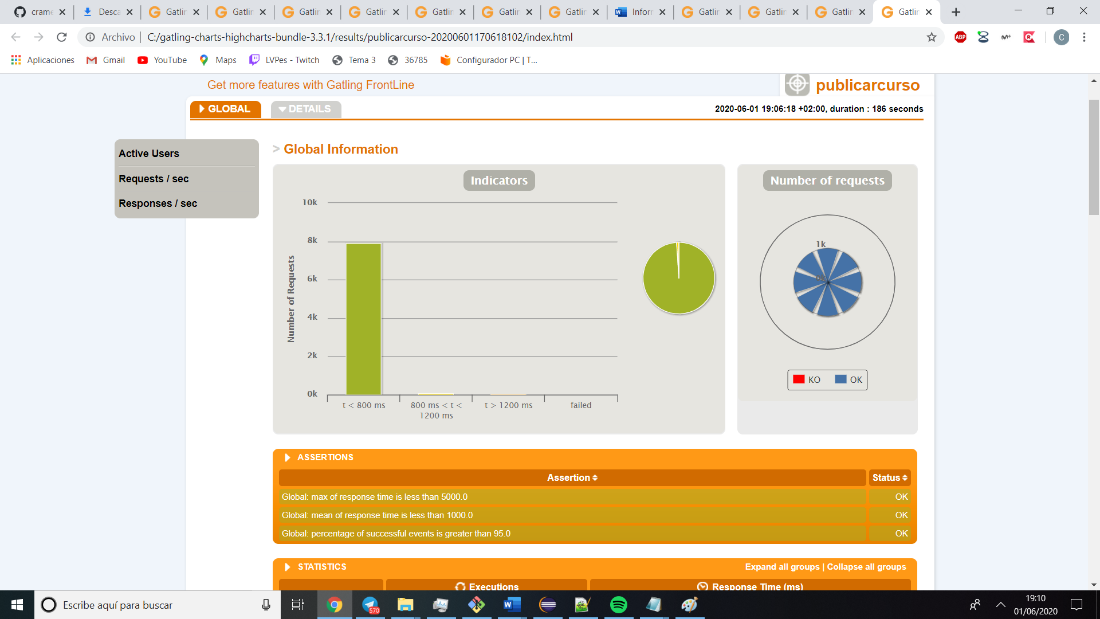


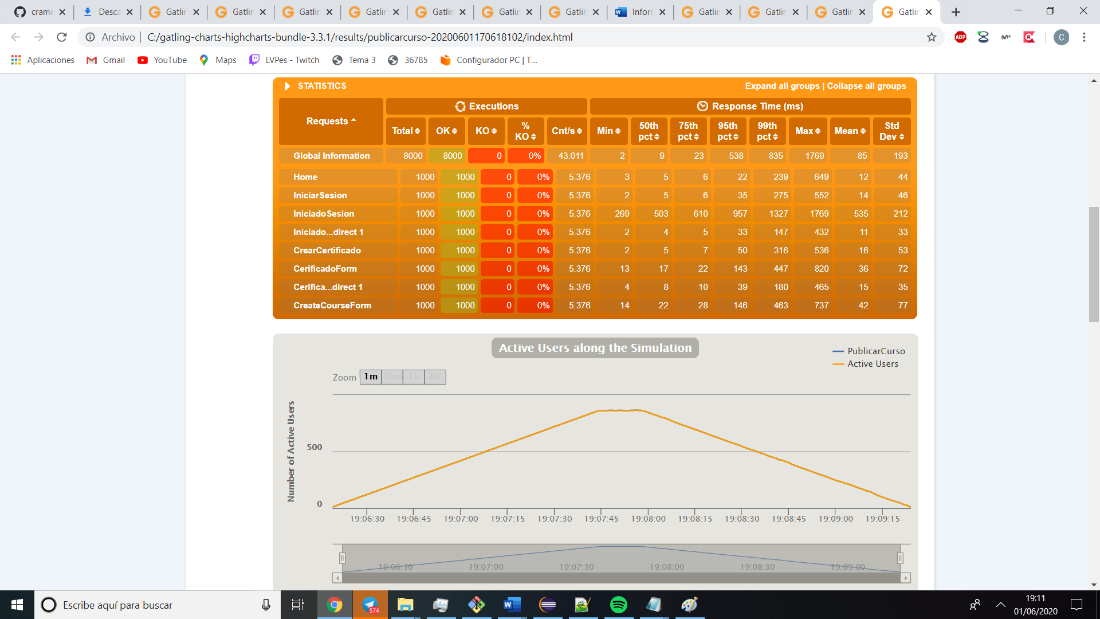


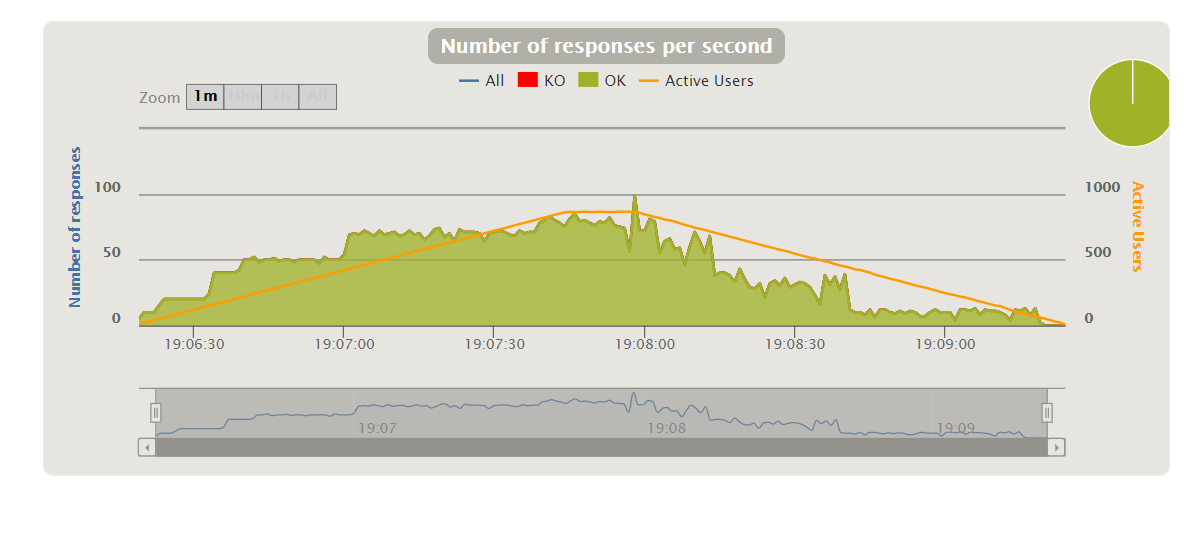


Esta captura fue realizada mientras se realizaba un ‘stress test’ de 1200 usuarios al sistema. Se puede identificar en la consola de Gatling que se producen errores de TimeOut en su mayoría, junto a otro error 500 que aparece menos. En el rendimiento del sistema se aprecia como a pesar de haber cargado ya todos los usuarios hay un pico del 100% (mencionar que al arrancar el test para cargar los usuarios se mantiene al 100%), dando lugar a un cuello de botella en la CPU. Cabe mencionar que se puede observar también el archivo index.html generado por Gatling para verificar que efectivamente el comportamiento es malísimo, dejando sin respuesta a un gran porcentaje de peticiones.

Una vez identificado el mínimo de usuarios que soporta este escenario (HU), se ha ido disminuyendo el número de usuarios concurrentes hasta llegar a diagnosticar el segundo parámetro solicitado, el máximo número de usuarios concurrentes para los cuales se obtiene un buen rendimiento por parte del sistema. En este con 1000 usuarios se realiza un rendimiento magnífico como se puede apreciar en la siguiente captura del index.html proporcionado por Gatling.







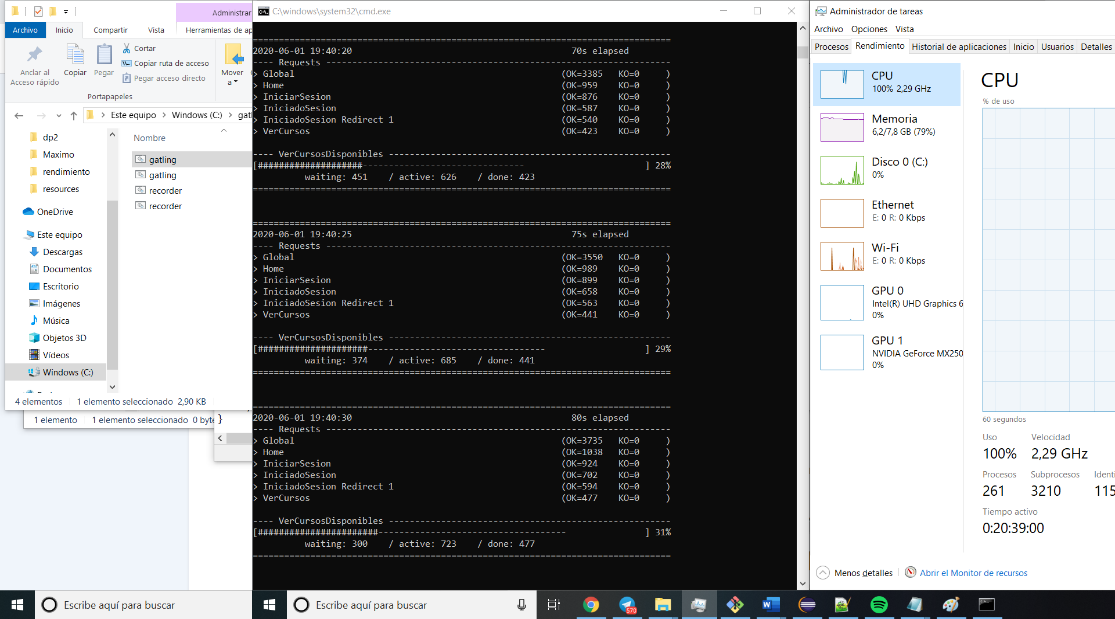
Se observa que satisface las 3 ‘assertions’ indicadas en el archivo .scala para la ejecución de este test y que todas las peticiones son resueltas de manera satisfactoria en un tiempo menor a 0,8 s (800ms)

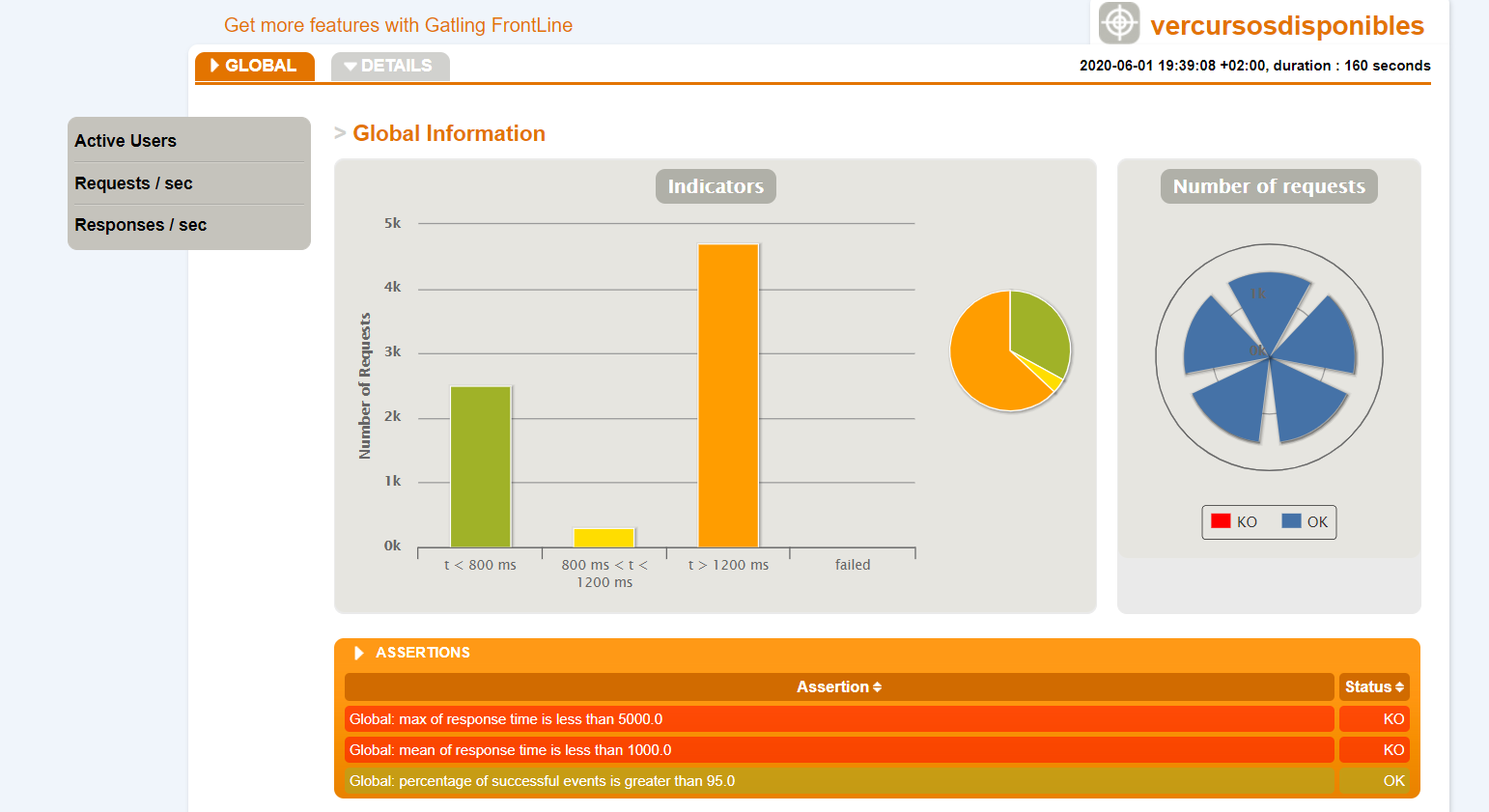
## HU7: Diagnosis

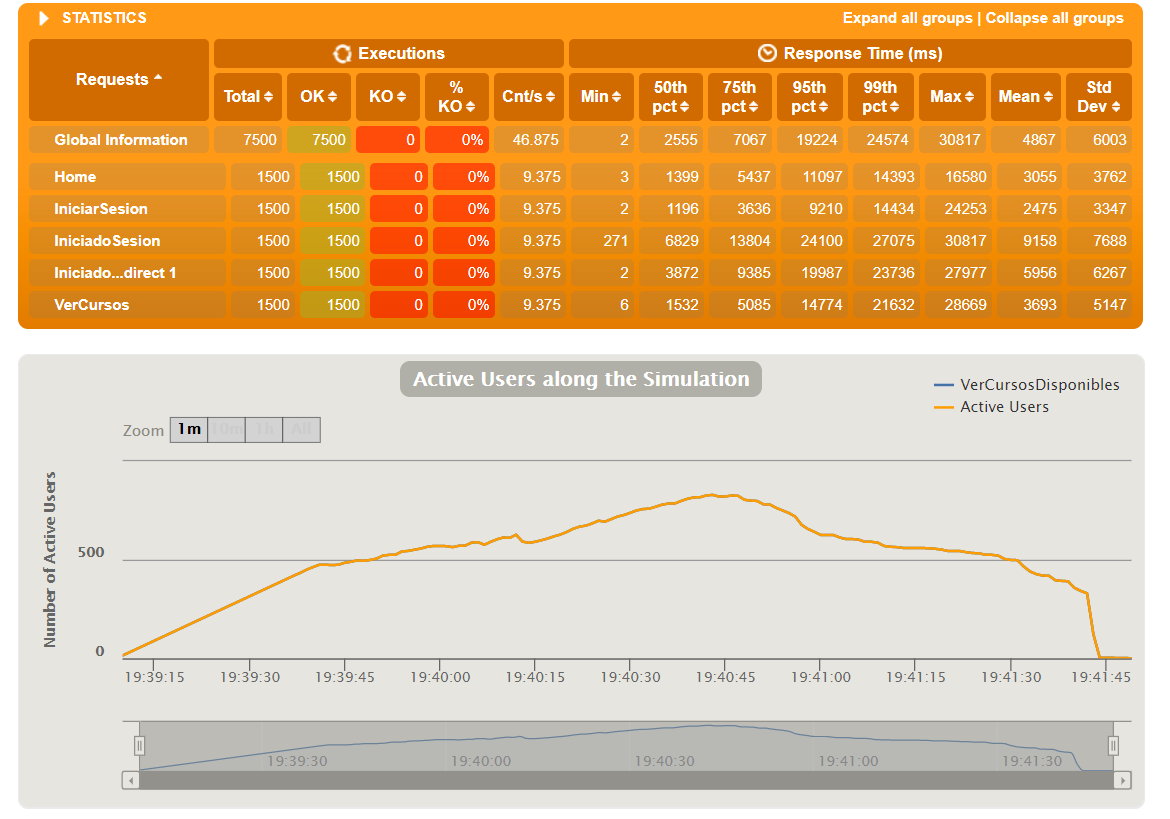
Este test de rendimiento se centra en la siguiente HU:

* El propietario de un animal debe poder listar los cursos disponibles en el sistema.

El primer parámetro a identificar es el mínimo de usuarios concurrentes que no puede soportar esta HU junto a una captura de pantalla sobre el rendimiento del sistema mostrando el cuello de botella en la CPU y/o RAM.



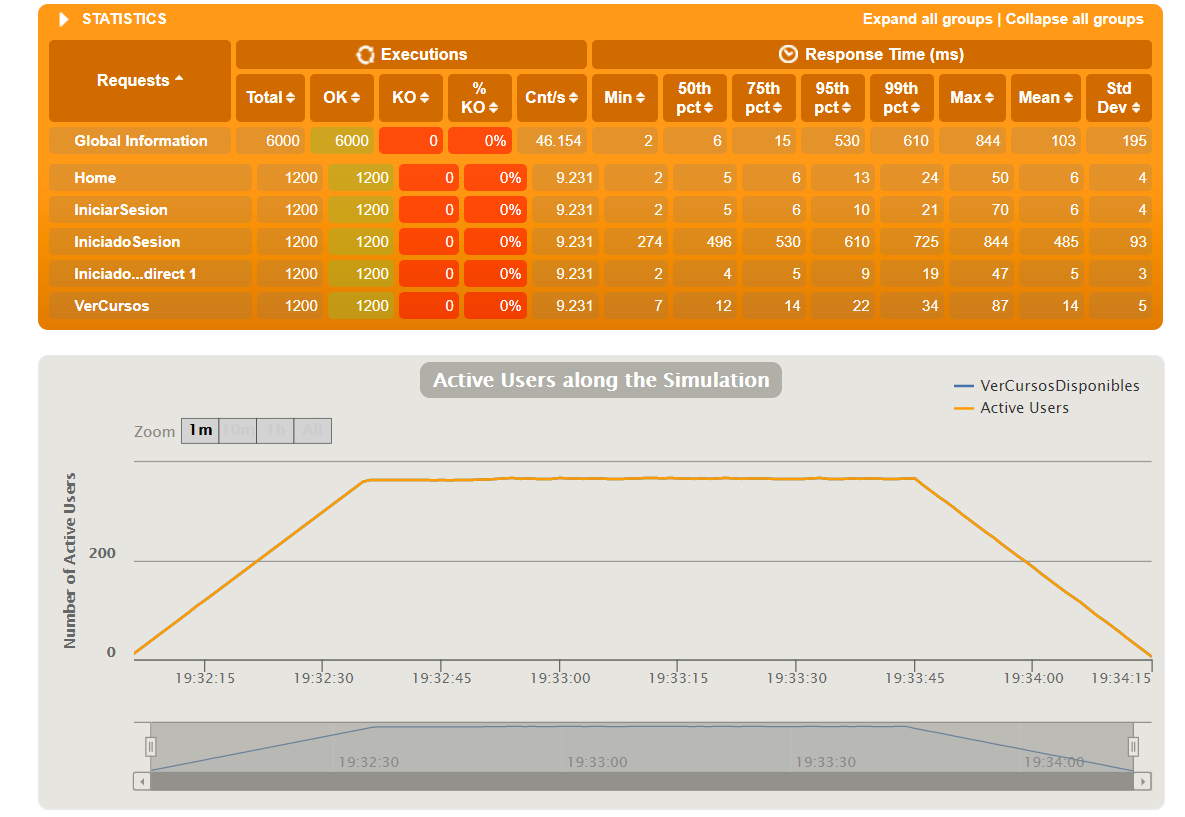


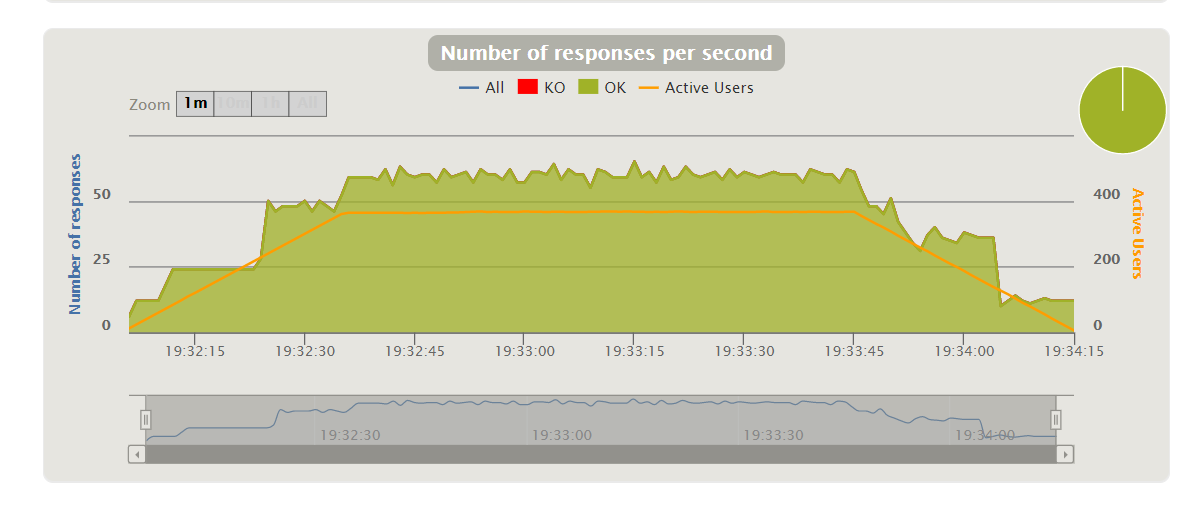


Esta captura fue realizada mientras se realizaba un ‘stress test’ de 1500 usuarios al sistema. En este caso no se han identificado ningún error, es decir, todas las peticiones han sido contestadas. Sin embargo, se ve como la CPU está saturada, manteniéndose durante todo el test en un 100%, dando lugar a un tiempo de respuesta nada óptimo. En esta situación el rendimiento no es malísimo, pero no es lo que queremos ya que en el reporte generado por Gatling se puede observar que el 63% de las peticiones tarda más de 1,2 s. Siendo éste un rendimiento bastante mejorable. No cumple las dos ‘assertions ’ del tiempo de respuesta.

Una vez identificado el mínimo de usuarios que soporta este escenario (HU), se ha ido disminuyendo el número de usuarios concurrentes hasta llegar a diagnosticar el segundo parámetro solicitado, el máximo número de usuarios concurrentes para los cuales se obtiene un buen rendimiento por parte del sistema. En este con 1200 usuarios se realiza un rendimiento magnífico como se puede apreciar en la siguiente captura del index.html proporcionado por Gatling.







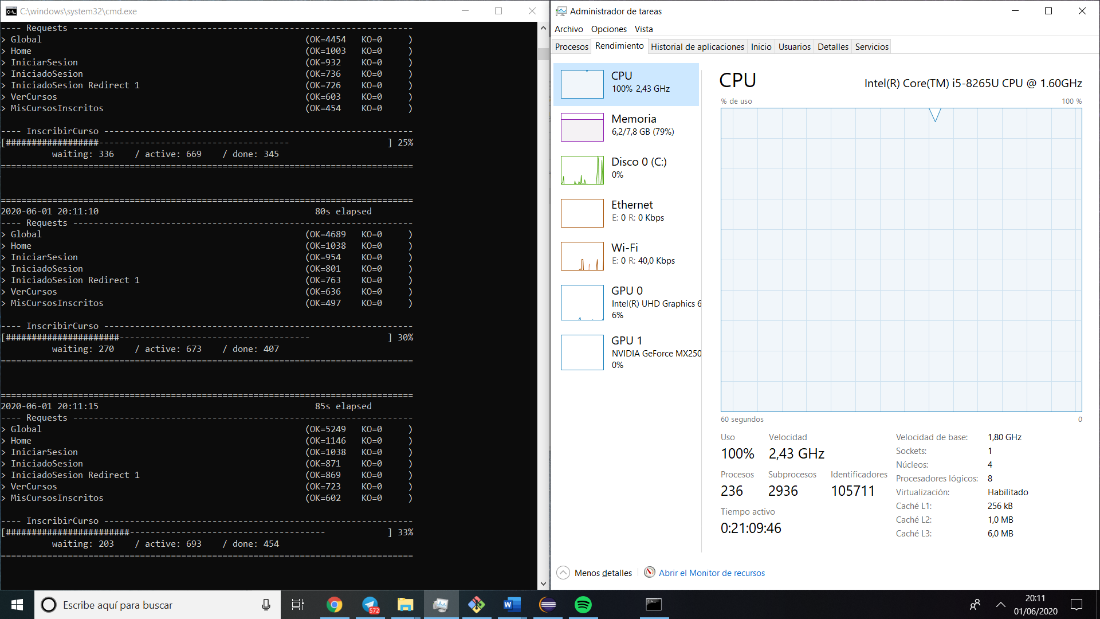
Se observa que satisface las 3 ‘assertions’ indicadas en el archivo .scala para la ejecución de este test y que todas las peticiones son resueltas de manera satisfactoria en un tiempo menor a 0,8 s (800ms).

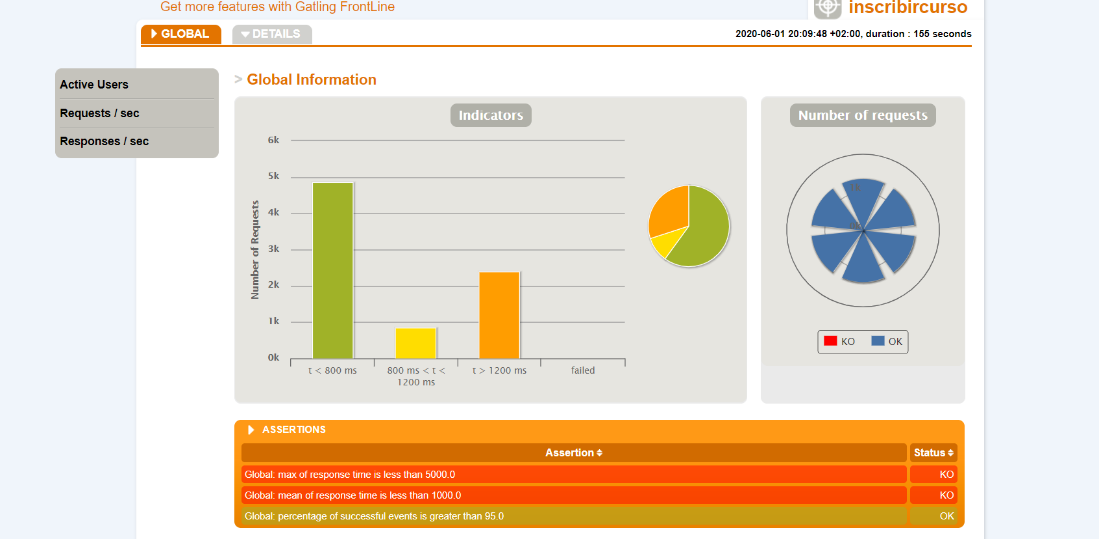
## HU8: Diagnosis

Este test de rendimiento se centra en la siguiente HU:

* El propietario de un animal debe poder inscribirse en un curso si no ha comenzado

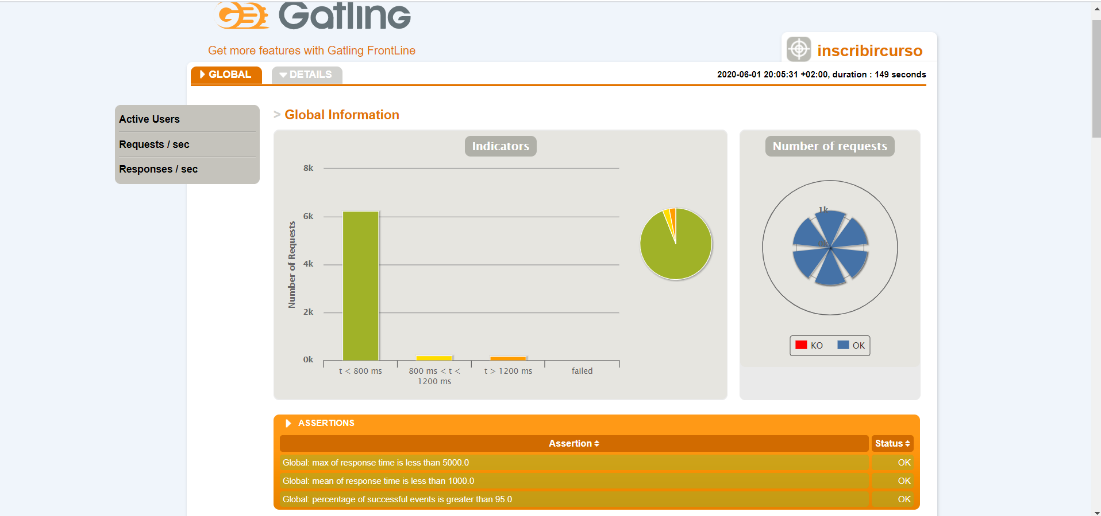
El primer parámetro a identificar es el mínimo de usuarios concurrentes que no puede soportar esta HU junto a una captura de pantalla sobre el rendimiento del sistema mostrando el cuello de botella en la CPU y/o RAM.

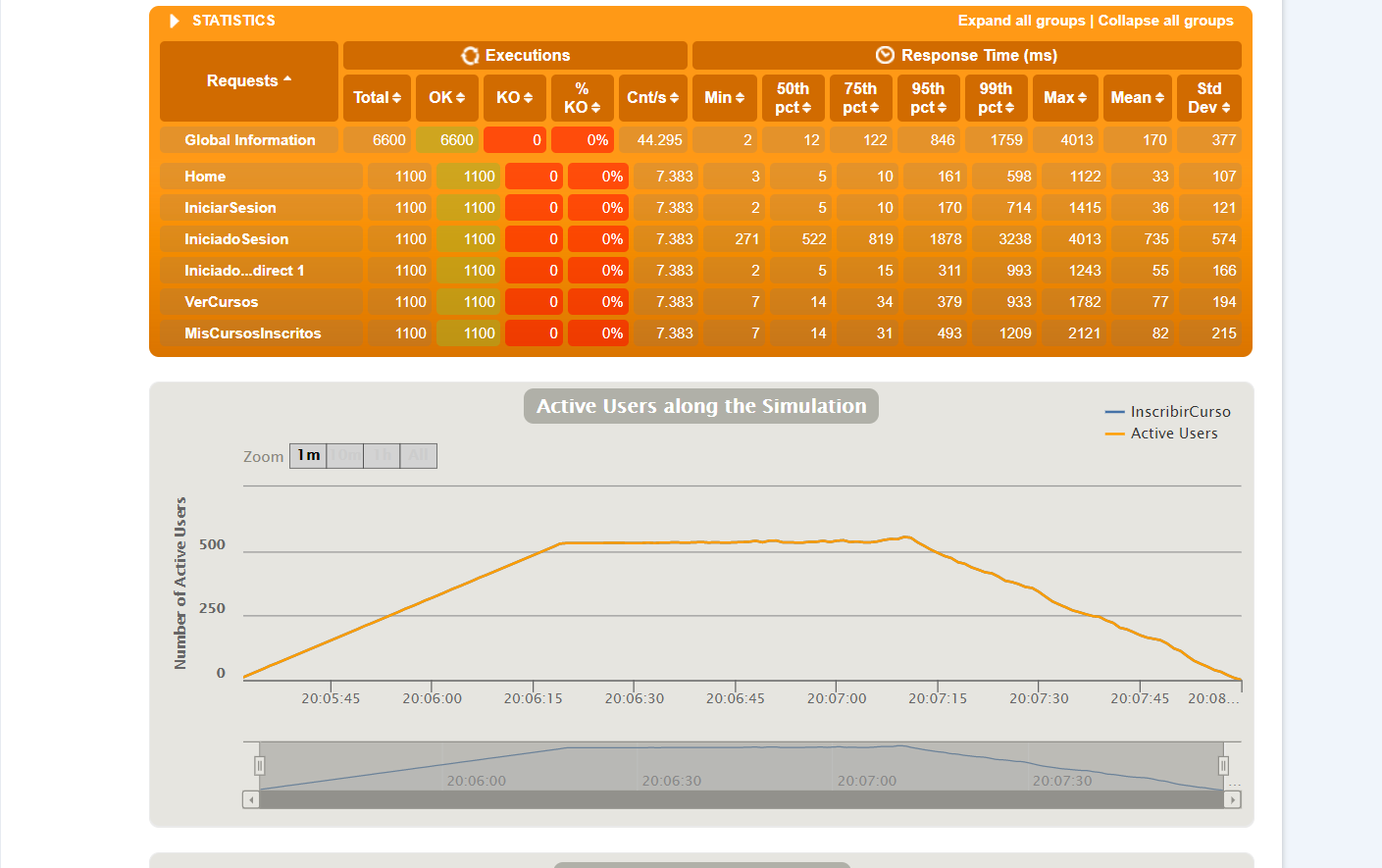




Esta captura fue realizada mientras se realizaba un ‘stress test’ de 1500 usuarios al sistema. En este caso no se han identificado ningún error, es decir, todas las peticiones han sido contestadas. Sin embargo, se ve como la CPU está saturada, manteniéndose durante todo el test en un 100%, dando lugar a un tiempo de respuesta nada óptimo. En esta situación el rendimiento no es malísimo, pero no es lo que queremos ya que en el reporte generado por Gatling se puede observar que el 30% de las peticiones tarda más de 1,2 s y un 10% tarda entre 0,8s y 1,2s. Siendo un éste un rendimiento bastante mejorable, ya que no se cumplen dos ‘assertions’ de Gatling.

Una vez identificado el mínimo de usuarios que soporta este escenario (HU), se ha ido disminuyendo el número de usuarios concurrentes hasta llegar a diagnosticar el segundo parámetro solicitado, el máximo número de usuarios concurrentes para los cuales se obtiene un buen rendimiento por parte del sistema.





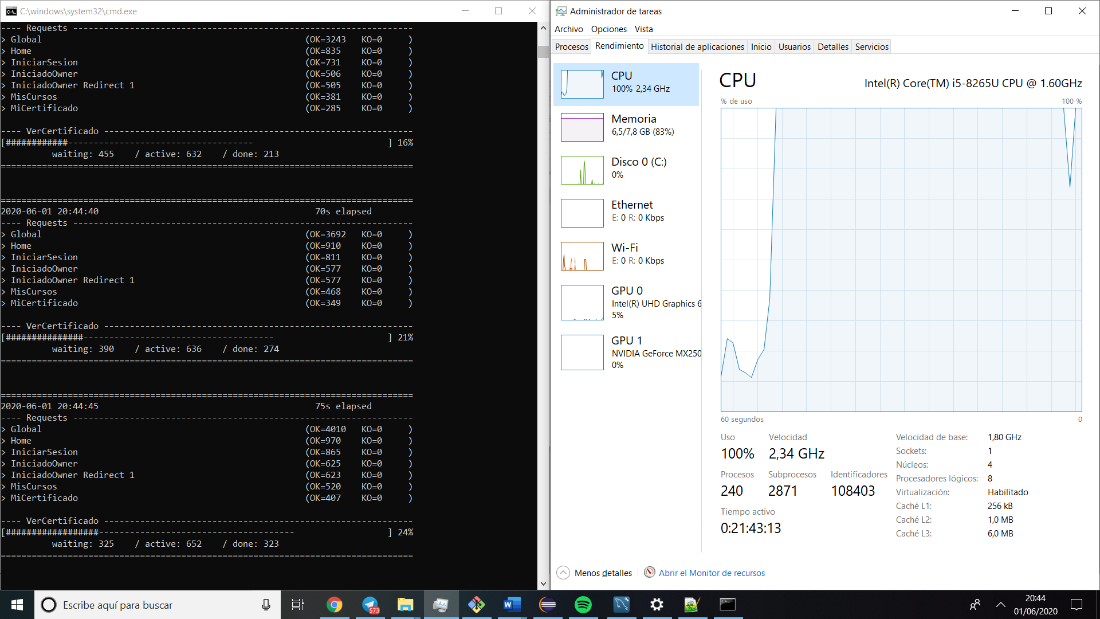
En este con 1100 usuarios se realiza un rendimiento que puede ser mejorable, pero sigue siendo bastante bueno ya que un 94% de las peticiones son respondidas en menos de 0,8s y se cumplen todas las ‘assertions’ de gatling.

## HU9: Diagnosis

Este test de rendimiento se centra en la siguiente HU:

* El propietario debe poder obtener un certificado del curso finalizado.

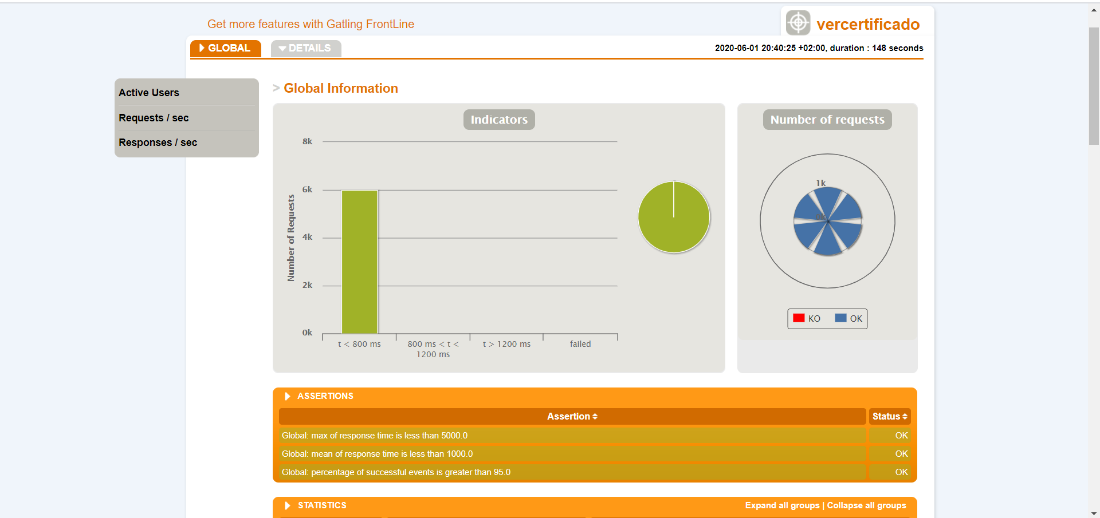
El primer parámetro a identificar es el mínimo de usuarios concurrentes que no puede soportar esta HU junto a una captura de pantalla sobre el rendimiento del sistema mostrando el cuello de botella en la CPU y/o RAM.

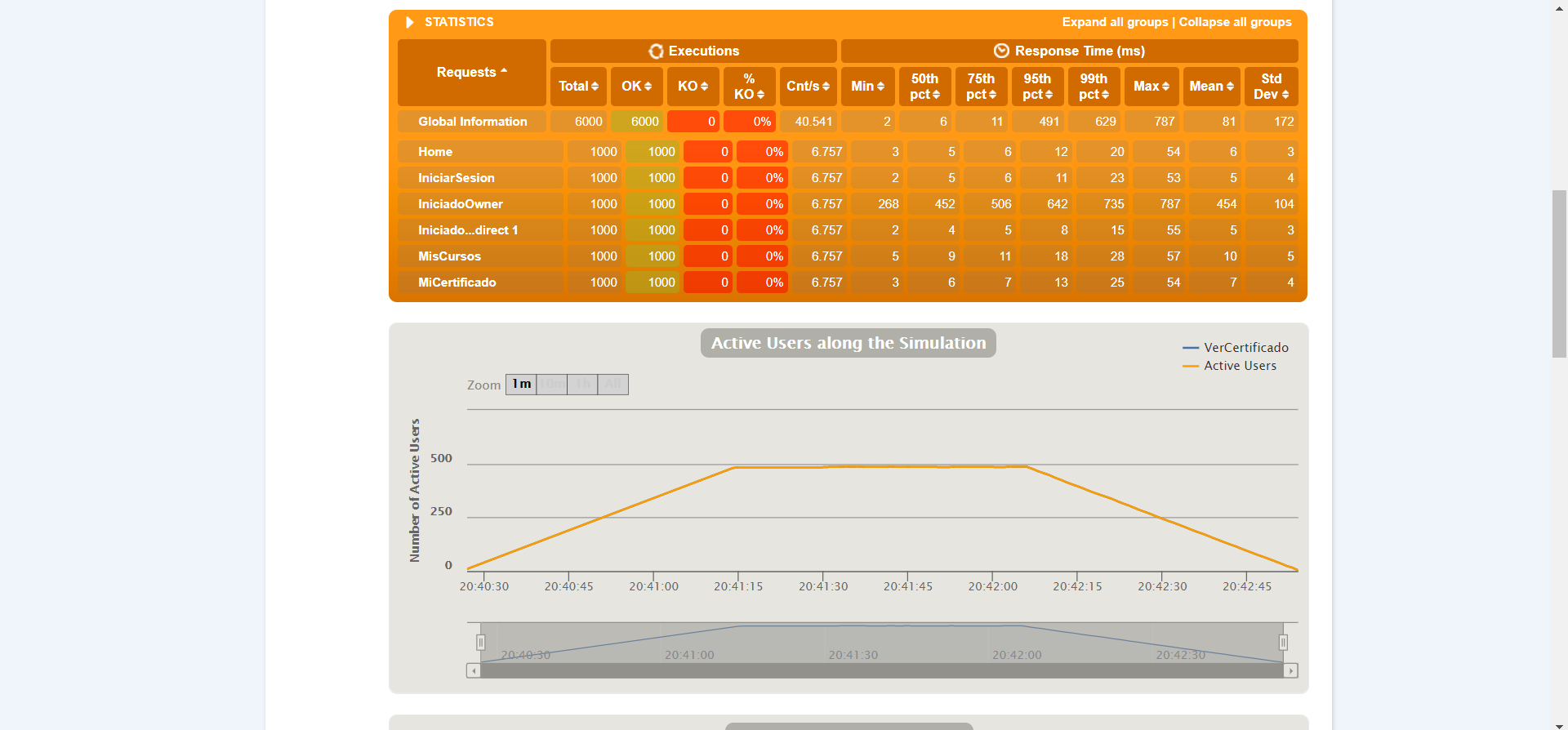




Esta captura fue realizada mientras se realizaba un ‘stress test’ de 1300 usuarios al sistema. En este caso no se han identificado ningún error, es decir, todas las peticiones han sido contestadas. Sin embargo, se ve como la CPU está saturada, manteniéndose durante todo el test en un 100%, dando lugar a un tiempo de respuesta nada óptimo. En esta situación el rendimiento no es malísimo, pero no es lo que queremos ya que en el reporte generado por Gatling se puede observar que el 34% de las peticiones tarda más de 1,2 s y un 9% tarda entre 0,8s y 1,2s. Siendo un éste un rendimiento bastante mejorable, ya que no se cumplen dos ‘assertions’ de Gatling.

Una vez identificado el mínimo de usuarios que soporta este escenario (HU), se ha ido disminuyendo el número de usuarios concurrentes hasta llegar a diagnosticar el segundo parámetro solicitado, el máximo número de usuarios concurrentes para los cuales se obtiene un buen rendimiento por parte del sistema.





En este con 1000 usuarios se realiza un rendimiento excelente con un 100% de las peticiones respondidas en menos de 0,8s.

## HU10: Diagnosis

Esta HU es la siguiente:

* El propietario de un animal exótico debe poder obtener un seguro al terminar el curso.

Esta HU, se ha considerado no someterle a test de rendimiento ya que es ver tus cursos en los que estás inscrito, pero con un atributo más dicha tabla. A nivel de rendimiento es exactamente lo mismo que la HU7, ya que los pasos son exactamente los mismos.

## HU11: Diagnosis

Esta HU es la siguiente:

* El propietario de un animal exótico, una vez realizado el curso debe poder consultar el seguro otorgado.

Esta HU, se ha considerado no someterle a test de rendimiento ya que los pasos son exactamente igual que la HU9. Teniendo además el mismo número de atributos la entidad Certificado que la entidad Seguro. Es lo mismo, únicamente que un seguro te aparecerá en el caso de que tengas una mascota exótica además de haber finalizado el curso de manera satisfactoria.

## HU12: Diagnosis

Esta HU es la siguiente:

* El propietario debe tener un descuento del 10% para siguientes inscripciones a cursos.

Esta HU, se ha considerado no someterle a test de rendimiento un motivo similar a la HU10. Ya que un propietario si ha finalizado dos cursos o más, le aparecerá un nuevo atributo en la tabla de los cursos disponibles (HU7) llamado precio rebajado.

## HU13: Diagnosis

Esta HU es la siguiente:

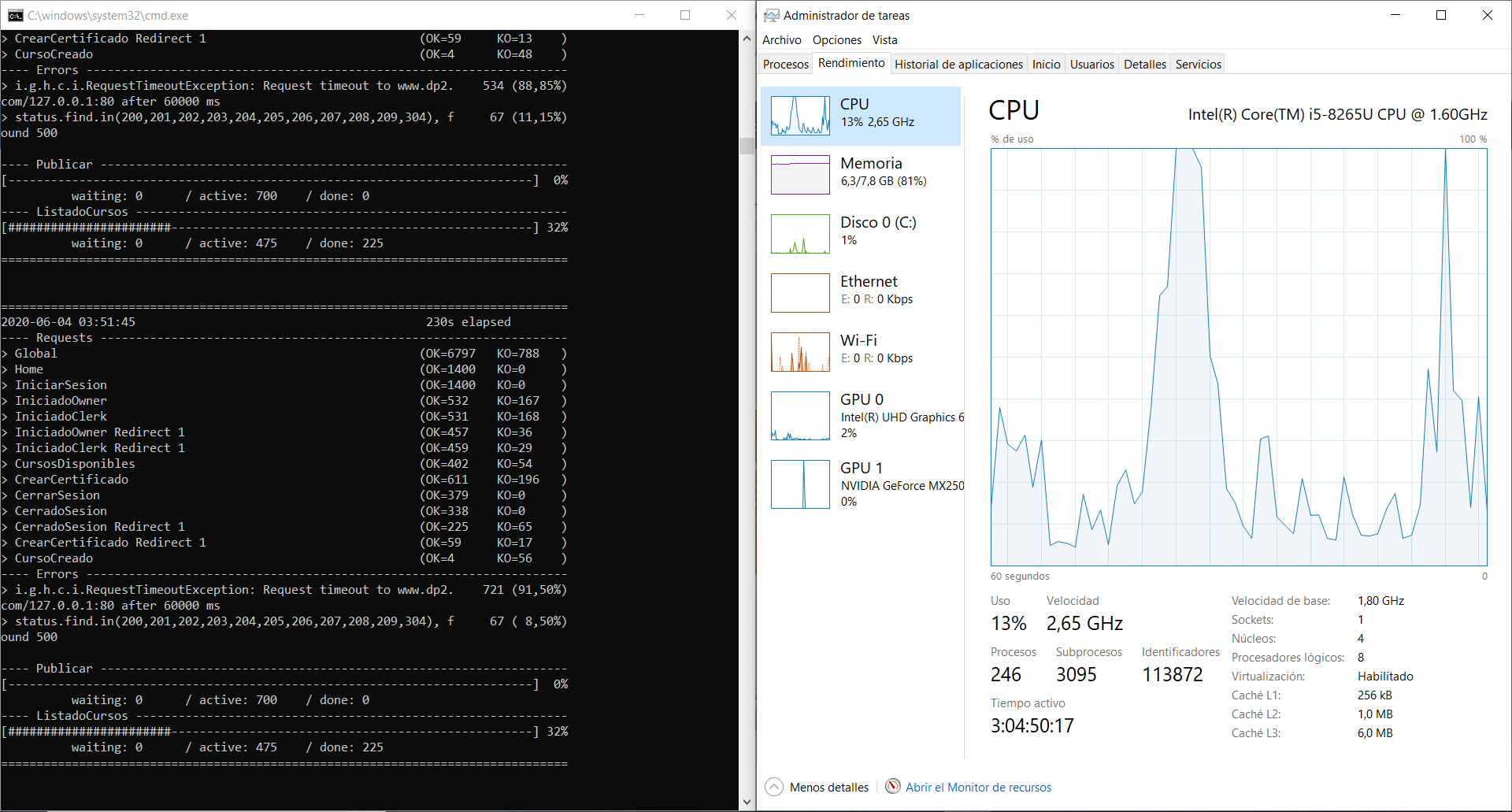
* El propietario debe poder ver toda la información de un curso antes de inscribirse.

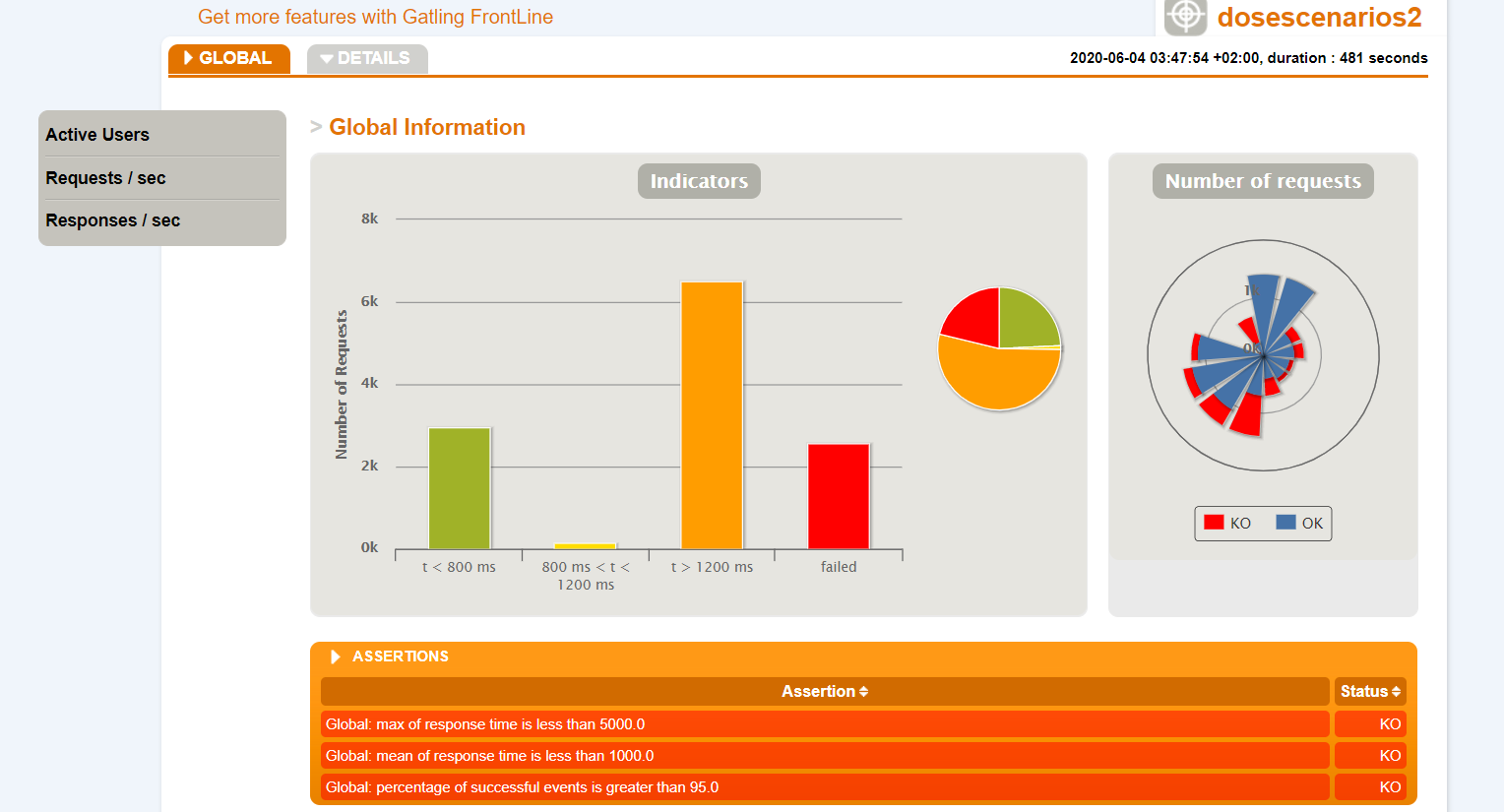
Esta HU, se ha considerado no someterle a test de rendimiento ya que en el listado de cursos disponibles aparece toda la información relativa a ese curso. Ya que este consta de muy pocos atributos.

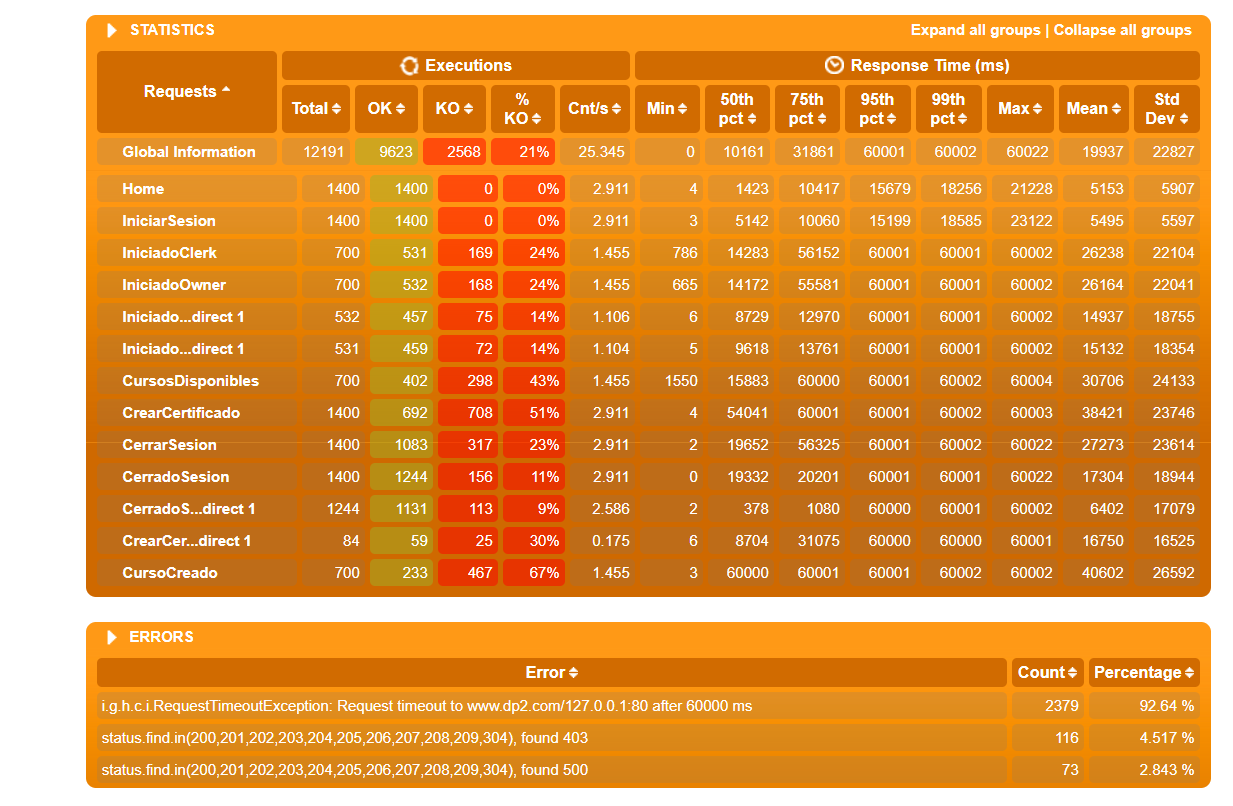
## HU6 vs HU8: Diagnosis

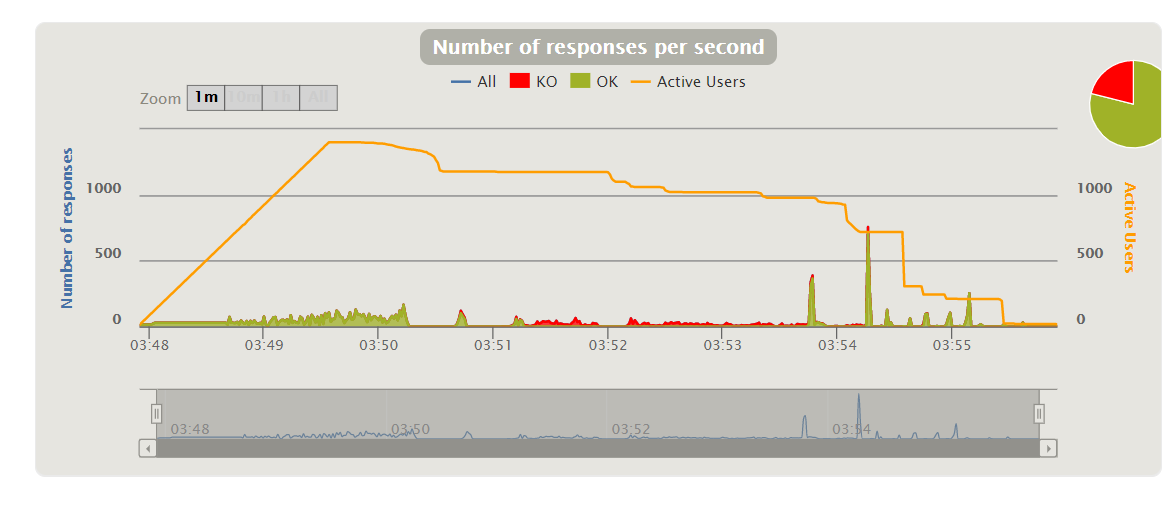
Este test se basa en comparar el rendimiento de la HU6 (Publicar un curso) frente a la HU8 (Inscribirse en un curso). Mencionar que la HU6 la realiza el rol Clerk y la HU8 la realiza el rol Owner.

El primer parámetro a identificar es el mínimo de usuarios concurrentes que no puede soportar esta HU junto a una captura de pantalla sobre el rendimiento del sistema mostrando el cuello de botella en la CPU y/o RAM.





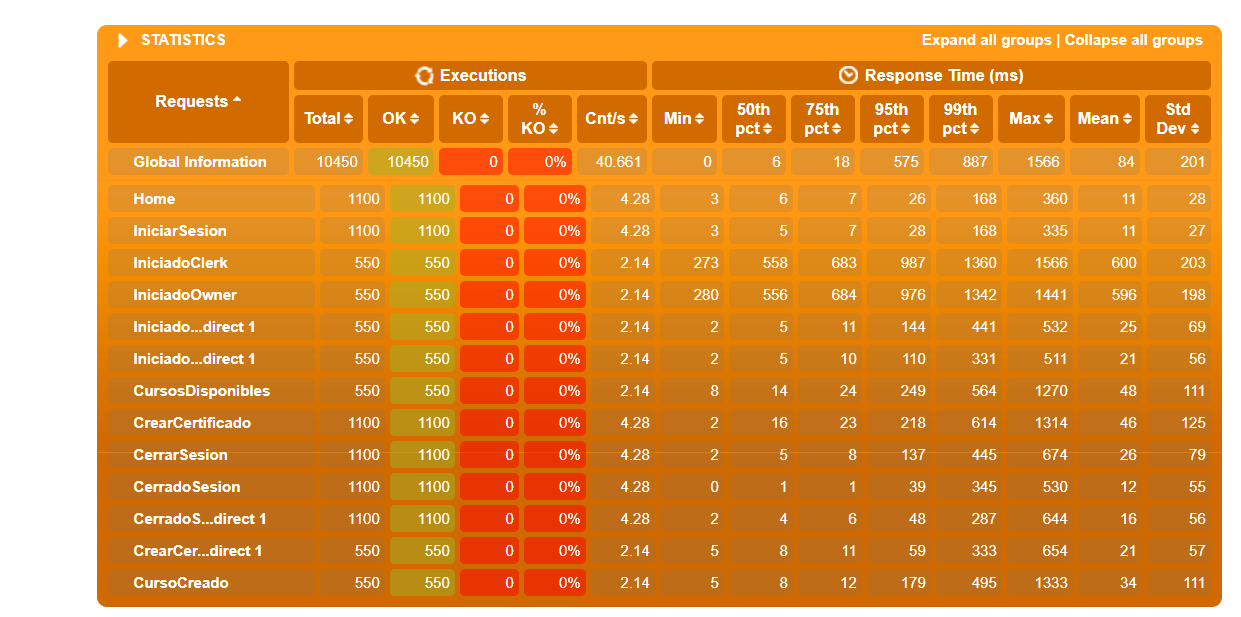


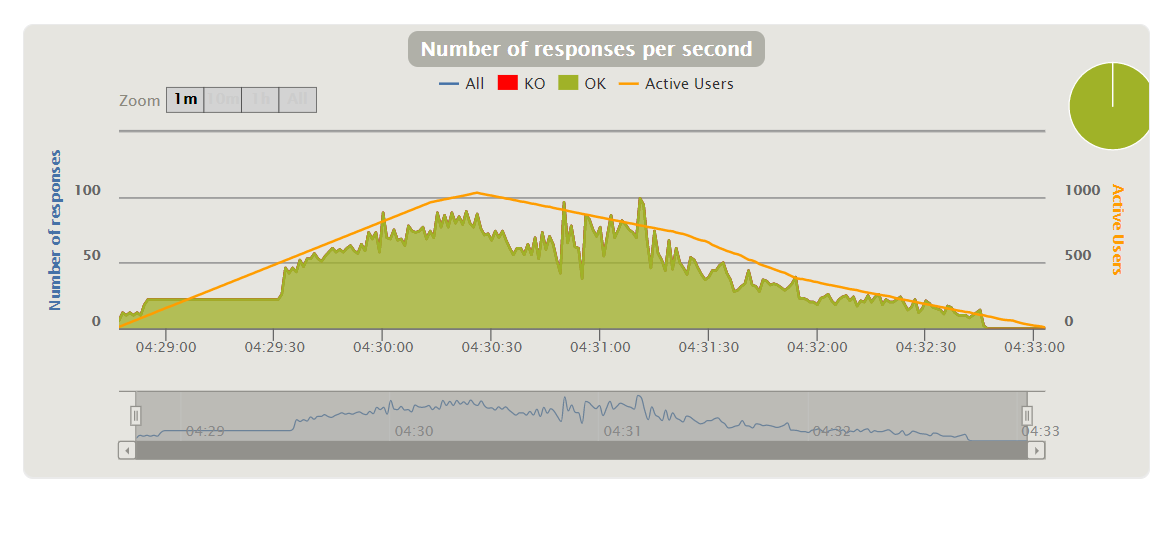


Esta captura fue realizada mientras se realizaba un ‘stress test’ de 700 usuarios por cada escenario al sistema (1400 en total). Se puede identificar en la consola de Gatling que se producen errores de TimeOut en su mayoría, junto a otro error 500 que aparece menos. En el rendimiento del sistema se aprecia como a pesar de haber cargado ya todos los usuarios hay un pico del 100% (mencionar que al arrancar el test para cargar los usuarios se mantiene al 100%), dando lugar a un cuello de botella en la CPU. Recordar que se puede observar también el archivo index.html generado por Gatling para verificar que efectivamente el comportamiento es malísimo, dejando sin respuesta a un gran porcentaje de peticiones.

Una vez identificado el mínimo de usuarios que soporta este escenario (HU), se ha ido disminuyendo el número de usuarios concurrentes hasta llegar a diagnosticar el segundo parámetro solicitado, el máximo número de usuarios concurrentes para los cuales se obtiene un buen rendimiento por parte del sistema.







En este con 550 usuarios cada escenario (1100 en total) se realiza un rendimiento excelente con un 100% de las peticiones respondidas en menos de 0,8s.

# Conclusión

En primer lugar, consideramos que para tener una visión objetiva de nuestro sistema a nivel de rendimiento se debería de haber hecho en un solo dispositivo, ya que dependiendo de las especificaciones donde se corran los test puede variar considerablemente. Aun así, es de mucha utilidad para identificar el comportamiento del sistema cuando se somete a mucho estrés. El máximo número de usuarios concurrentes que soporta nuestra aplicación es de 560. Con este número de usuarios garantizamos que más del 95% de las peticiones se satisfacen y que la media del número de peticiones totales no superan el segundo, dando lugar a un funcionamiento más que correcto.

Por último, se han realizado pruebas en el que los usuarios concurrentes son introducidos en un corto periodo de tiempo, es decir bajando de esos 100 segundos. Esto ha significado para el sistema una bajada de rendimiento muy considerable, ya que el sistema debe satisfacer más peticiones en un periodo de tiempo más corto dando a lugar a un cuello de botella grande en las peticiones que interactúen con la base datos provocando TimeOuts. Es digno mencionar lo interesante que nos ha parecido la herramienta Gatling ya que teníamos experiencia con Jmeter en la cual hay una serie de mejoras sustanciales como la fácil configuración y la paralelización de peticiones proporcionando una mayor velocidad de ejecución de todo el test. También proporciona un “report” muy completo donde se puede realizar un análisis de manera sencilla y verificar con “assertions” posibles cuellos de botellas y posibles mejoras de rendimientos en métodos concretos.