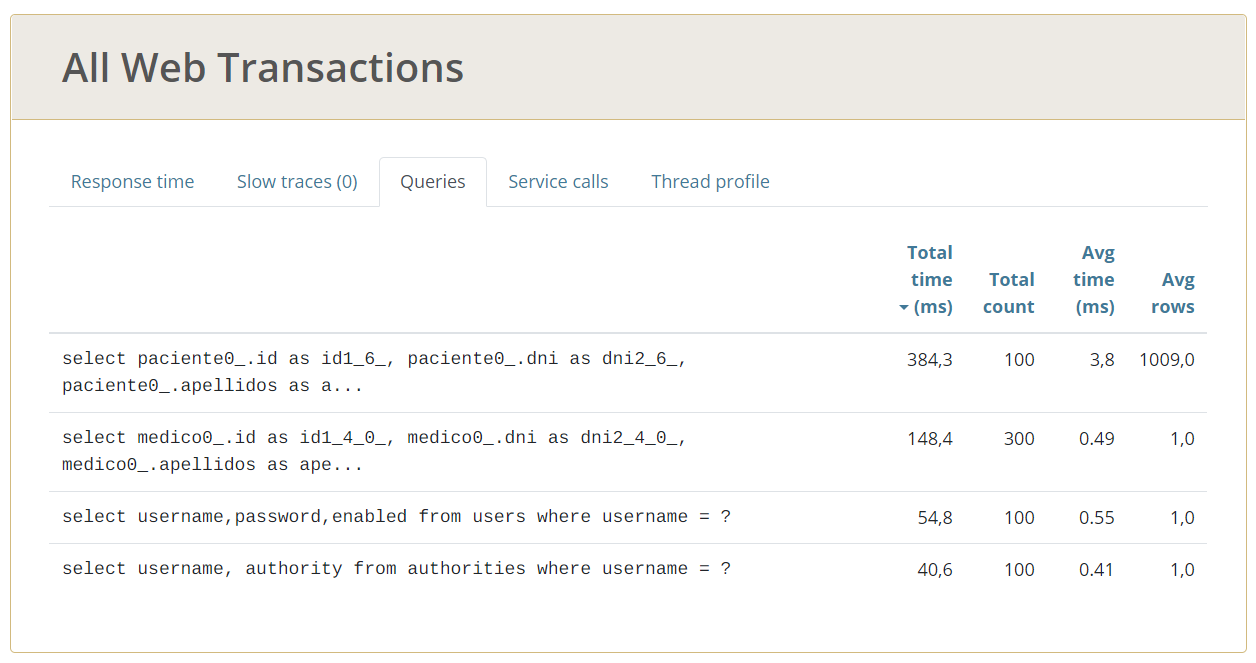
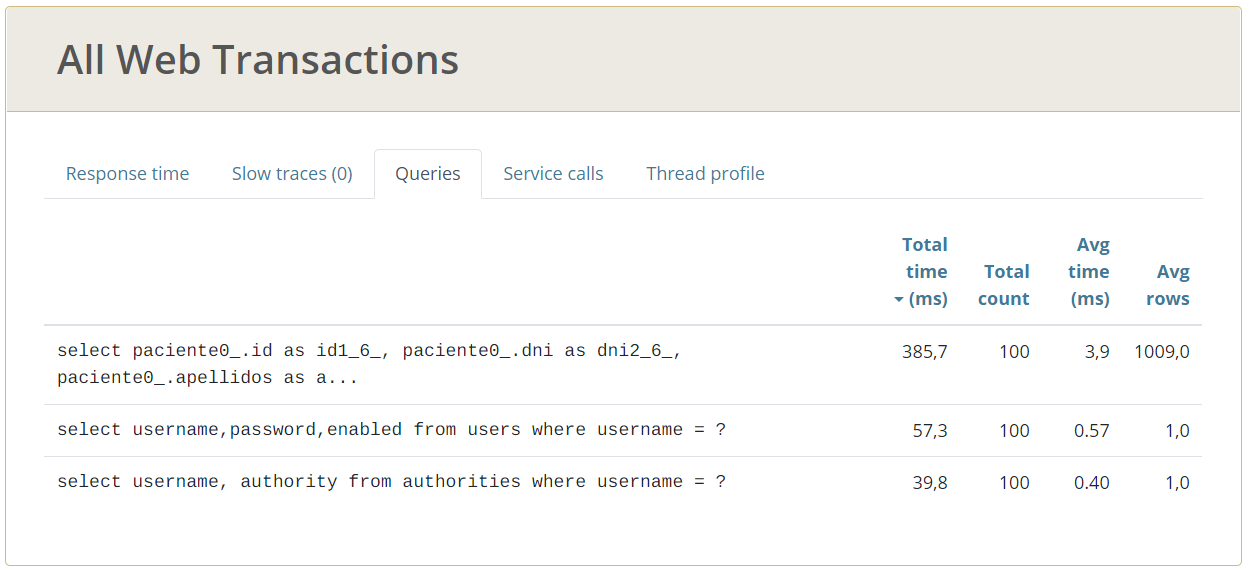
**Profiling y Optimización**

A continuación se detallará el profiling realizado a la Historia de Usuario 4, correspondiente a listar todos los pacientes del sistema. Esta funcionalidad parece tener problemas de rendimiento cuando existen un número elevado de pacientes dentro del sistema, el cual dado la naturaleza de este, no parará de crecer día tras día. Siendo esta una funcionalidad elemental en el proyecto y que será usada por todos los médicos registrados, resulta importante realizar un profiling para optimizar su rendimiento.

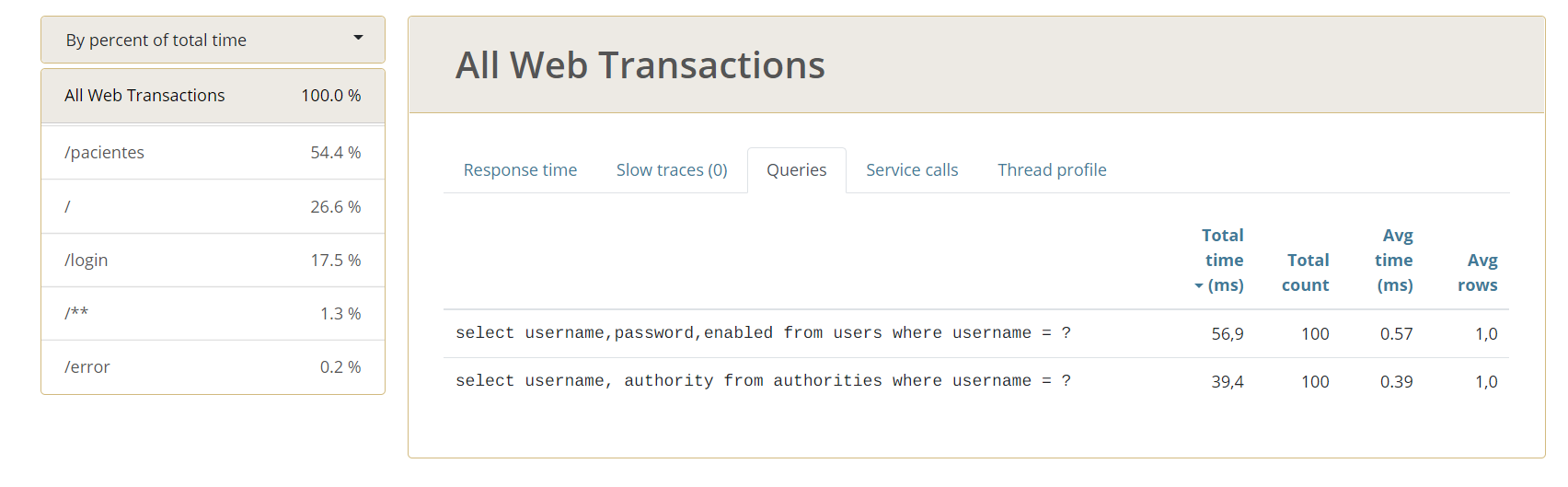
Así pues, se procedió a realizar un análisis de dicha funcionalidad con Glowroot insertando previamente 1.000 pacientes en el sistema con Gatling, un número que llegará a tener probablemente en no demasiado tiempo un sistema cuyo propósito principal es el almacenamiento de pacientes y datos relacionados con ellos. De este modo obtuvimos el siguiente informe de parte de Glowroot:



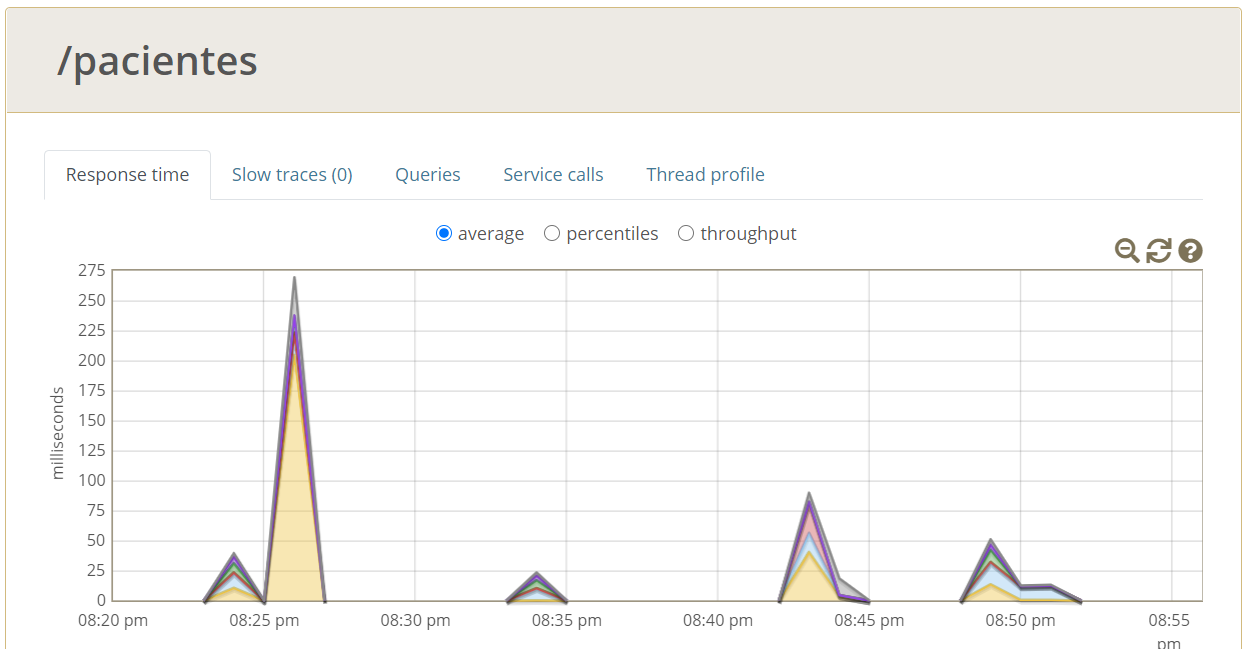
En esta imagen podemos observar las querys realizadas en el sistema de parte de 100 médicos que desean obtener la lista de pacientes registrados. Para empezar, ignoraremos las dos últimas querys pues pertenecen al logueo de los médicos del sistema. Así pues, observamos que las dos primeras querys pertenecen a la historia de usuario que queremos analizar. Podemos observar que se realizan dos querys: una correspondiente a los pacientes y otra correspondiente a los médicos. Además observamos un problema significativo: se realizan el triple de consultas correspondiente a médicos que a pacientes. Esto tiene aún menos sentido si tenemos en cuenta que el listado general de pacientes ni siquiera muestra el médico asociado a cada uno de ellos. Por otra parte, observamos que el tiempo total que conlleva la consulta respectiva a los médicos es casi un tercio del tiempo total de consultas realizadas por esta funcionalidad. En definitiva, concluímos que podríamos mejorar el rendimiento de manera significativa si eliminábamos la consulta respectiva a los médicos que, en este caso, no tenía ningún propósito. Para ello, añadimos en el modelo de la clase paciente junto a *@ManyToOne* lo siguiente: *(fetch = FetchType.LAZY)*. De este modo conseguiremos que no se realice una consulta en base al médico del paciente cuando esta no sea necesaria. Con ello, volvimos a ejecutar el test y obtuvimos la siguiente información de Glowroot:



Podemos observar que las consultas respectivas a los médicos han desaparecido, reduciendo el tiempo total casi un tercio del total que obtuvimos previamente. Aún así en una funcionalidad tan importante y que irá aumentando tanto su tamaño según pase el tiempo, pues se irán añadiendo pacientes diariamente, resulta conveniente que su rendimiento sea el mejor posible. Por esta razón, pensamos que la implementación de caché para mejorar aún más su optimización resultaba una decisión beneficiosa para el sistema. De este modo, una vez realizada la implementación de caché para esta funcionalidad se volvieron a ejecutar los mismos tests y el resultado proporcionado por Glowroot fue el siguiente:



En esta imagen observamos que gracias a la caché, las querys han desaparecido. En la parte izquierda de la imagen podemos confirmar que se han realizado las consultas en cuestión, pues se ha accedido a la dirección correcta (*/pacientes*). Además se ha optado por introducir 120 segundos de tiempo de caché, dado que mantener la lista de pacientes actualizada es importante en nuestro sistema, pero este tiempo no causará inconvenientes a los usuarios que lo usen. De este modo concluímos el profiling de esta historia de usuario, habiendo alcanzado un rendimiento mucho mejor que el inicial. En la siguiente imagen podemos observar la evolución del rendimiento en base a los cambios realizados:



Si ignoramos los picos más pequeños de esta imagen (los cuales se corresponden a pruebas individuales para comprobar el correcto funcionamiento del sistema), podemos observar claramente tres picos significativos en la línea temporal. Cada uno de ellos se corresponden al código inicial y a los dos cambios realizados respectivamente. Así pues podemos concluir que cada uno de los cambios implantados ha conllevado un mejor rendimiento del sistema, finalizando en un rendimiento gratamente superior al que poseíamos inicialmente.