****

**SISTEMAS DISTRIBUIDOS**

**DOCUMENTACIÓN PROYECTO II PARCIAL - II TÉRMINO 2017**

**INTRODUCCIÓN**

La latencia es un valor que se relaciona directamente con la arquitectura de un sistema. Si este no se encuentra bien implementado el tiempo de respuesta para las solicitudes de los clientes que desean comunicarse con los servidores de nuestra aplicación web será alto, generando así inconformidades en los usuarios al no cumplir con las características de un sistema ágil y eficiente que es lo que usualmente se espera. Este proyecto implementa un sistema web, con una arquitectura de microservicio y una caché como solución para reducir la latencia de acceso a la base de datos. El sistema permite ver un top 10 de las noticias más populares de un periódico. Estas noticias se encuentras alojadas en una base de datos, y son leídas y seleccionadas por un microservicio que se encarga de generar el top 10 y al mismo tiempo verifica si se encuentran en caché, y de no ser así, procede a guardarlas para una futura consulta. El usuario interactúa con el sistema a través de una página web sencilla que cuenta con un solo botón, que al pulsarlo genera los resultados de top mostrándolos en una tabla.

**ESTRUCTURA DE DATOS**

Este proyecto fue desarrollado en lenguaje JAVA y se implementaron las siguientes clases:

* **Noticia**

Contiene los atributos y métodos que tiene una noticia leída de la base de datos.

* Atributos: Identificador, número de visitas, título y descripción.
* Métodos: getters y setters de los atributos, compareTo() sobreescrito, para ordenar los objetos de mayor a menor de acuerdo a número de visitas y método toString() sobreescrito para mostrar la información que contiene una noticia.
* **Conexión**

Establece la conexión del Servidor de Microservicios con la Base de Datos mediante el puerto y con la caché mediante los puertos 3306 y 11211 respectivamente.Los principales métodos que implementa esta clase son ObtenerTop10() y guardarCaché().

* LinkedList<Noticia>ObtenerTop10(): Retorna una lista con el top 10 de las noticias más visitadas que se encuentran almacenadas en la base de datos.
* LinkedList<Noticia>guardarCaché(): Verifica si dentro de la caché se encuentra cargado el top 10, de ser así lo retorna caso contario lo almacena mediante método set<> . Se debe especificar el tiempo en el que esta información permanecerá alojada.
* **HiloServidor**

Clase que implementa la interfaz Runnable de modo que se ejecute de manera concurrente. Prepara la comunicación del Servidor de Microservicio con el Reverse Proxy habilitando el Puerto 7911.

* **MicroServicio**

Esta es nuestra Clase principal debido a que realiza instancias de los objetos noticias, conexión e Hiloservidor, que interactúa entre sí para establecer una correcta conexión de los servidores y obtendremos el top 10 de las Noticias más visitadas alojadas en la base de datos.

* **Servidor**

Clase generada por ApacheThrift y adjuntada al programa de la aplicación del Microservicio

**MIDDLEWARE**

* APACHE THRIFT

Como middleware de nuestra red de miroservicio se usó Apache thrift debido a que se debía establecer una conexión desde el cliente Web con nuestra aplicación implementada en Java y este trabaja con varios lenguajes de programación, entre ellos Java y php. Otra ventaja de ApacheThrift es que la manera de comunicación es binaria debido que genera librerías binarias tanto para el servidor de microservicio (java) como el cliente web (php) y que deben ser adjuntadas en la parte del cliente.

**AWS (Amazon Web Service)**

Amazon Web Services (AWS) es una plataforma de servicios de [nube](https://aws.amazon.com/es/what-is-cloud-computing/) que ofrece potencia de cómputo, almacenamiento de bases de datos, entrega de contenido y otra funcionalidad para ayudar a las empresas a escalar y crecer. Explore cómo millones de [clientes](https://aws.amazon.com/es/solutions/case-studies/) aprovechan los [productos](https://aws.amazon.com/es/products/) y [soluciones](https://aws.amazon.com/es/solutions/) de la nube de AWS para crear aplicaciones sofisticadas y cada vez más flexibles, escalables y fiables.



**PRUBAS DE RENDIMIENTO**

* APACHE JMETER

Para llevar a cabo pruebas de latencia y trouhput en el sistema se usó Apache jMeter, un software libre desarrollado en Java que permite evaluar y medir el desempeño de aplicaciones web enviando un determinado número de peticiones a su servidor en un tiempo establecido. Provee herramientas útiles como Summary Report, Visualizar resultados en tablas y en árboles, Gráficas de Resultados que permiten analizar parámetros como, el estado de la petición (si fue o no exitosa), latencia, tiempo de conexión, bytes enviados, entre otros y adicionalmente nos muestra detalles de cómo el nombre tipo del servidor y su versión, fecha, tipo de contenido en la página web, etc.

A continuación, se mostrarán los resultados que se obtuvieron al realizar pruebas con dos escenarios distintos:

1. **Microservicio usando solamente la Base de Datos**

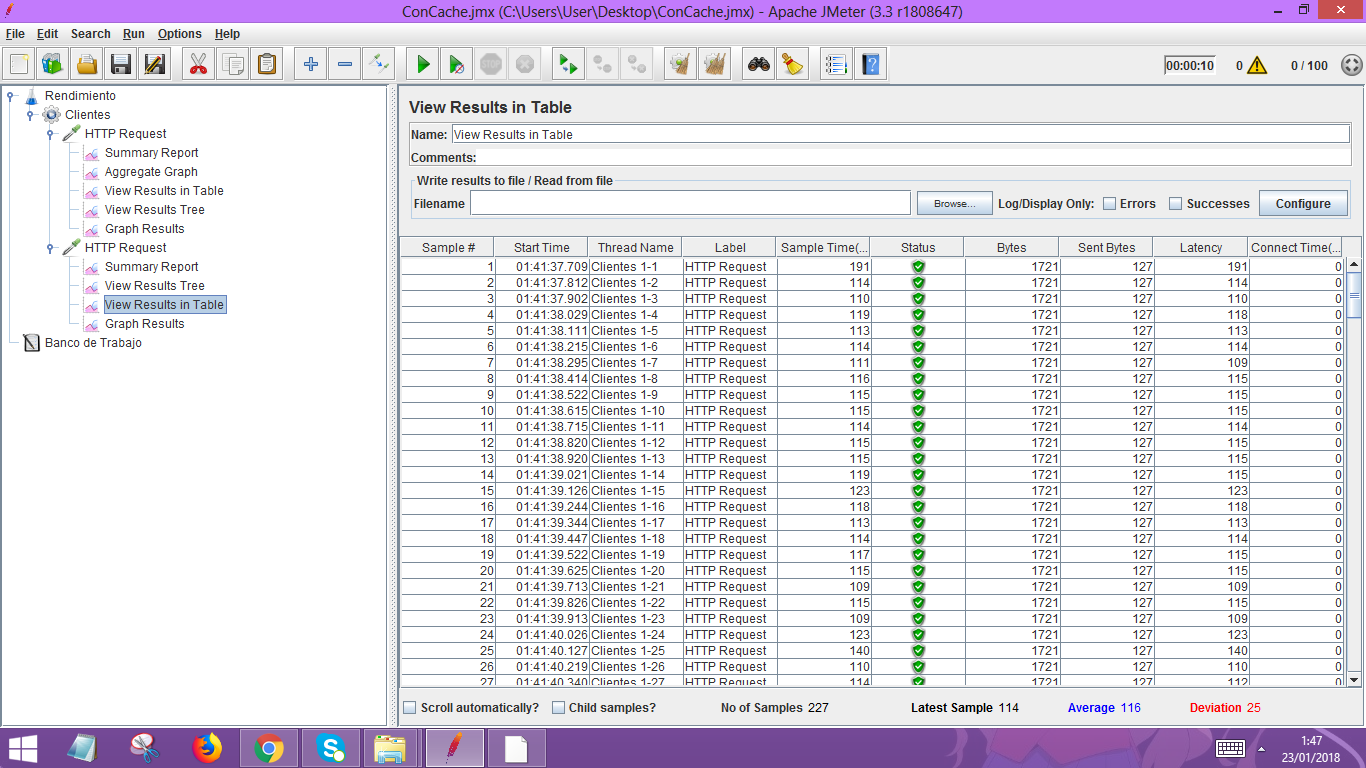


Ilustración . Resultados de la prueba de rendimiento A) presentados en una tabla

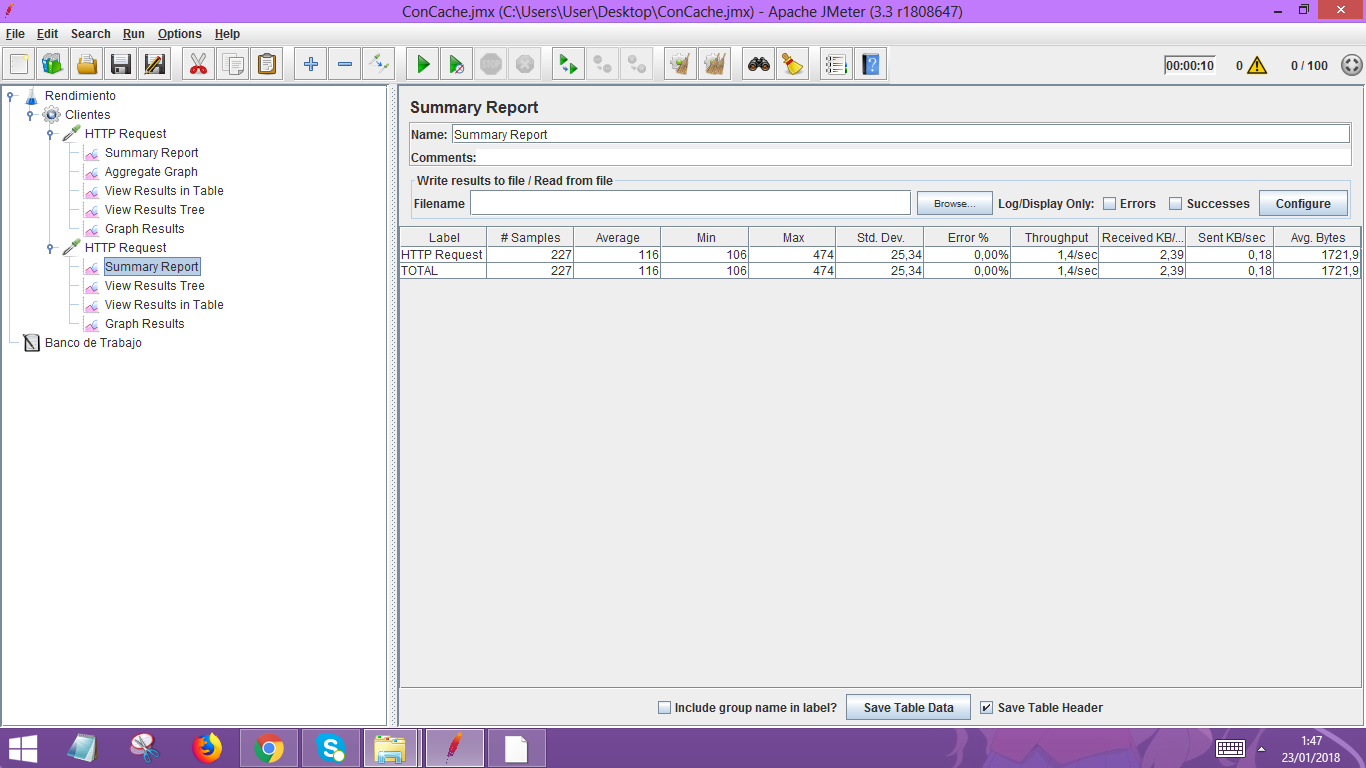


Ilustración . Resultados de la prueba de rendimiento A) presentados en un Summary Report (Se observa un valor de promedio de Througput de 1,4/seg )

Se puede observar que con 100 solicitudes al servicio web ejecutadas en un intervalo de 10 segundos, todas son respondidas satisfactoriamente pero el servidor se demora un poco en responder. El valor de throughput promedio obtenido es de 1,4 solicitudes/seg.

1. **Microservicio usando la caché más la Base de Datos.**

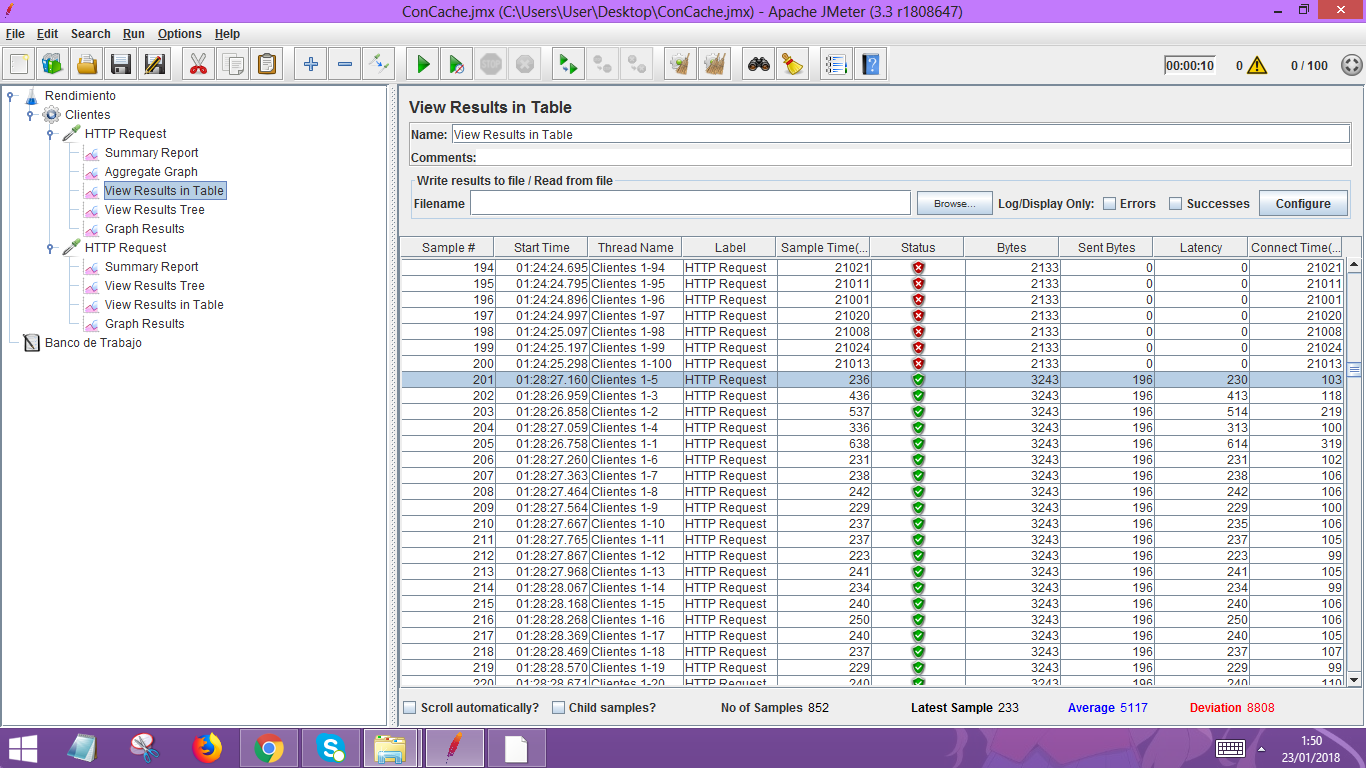


Ilustración .Resultados de la prueba de rendimiento B) presentados en una tabla

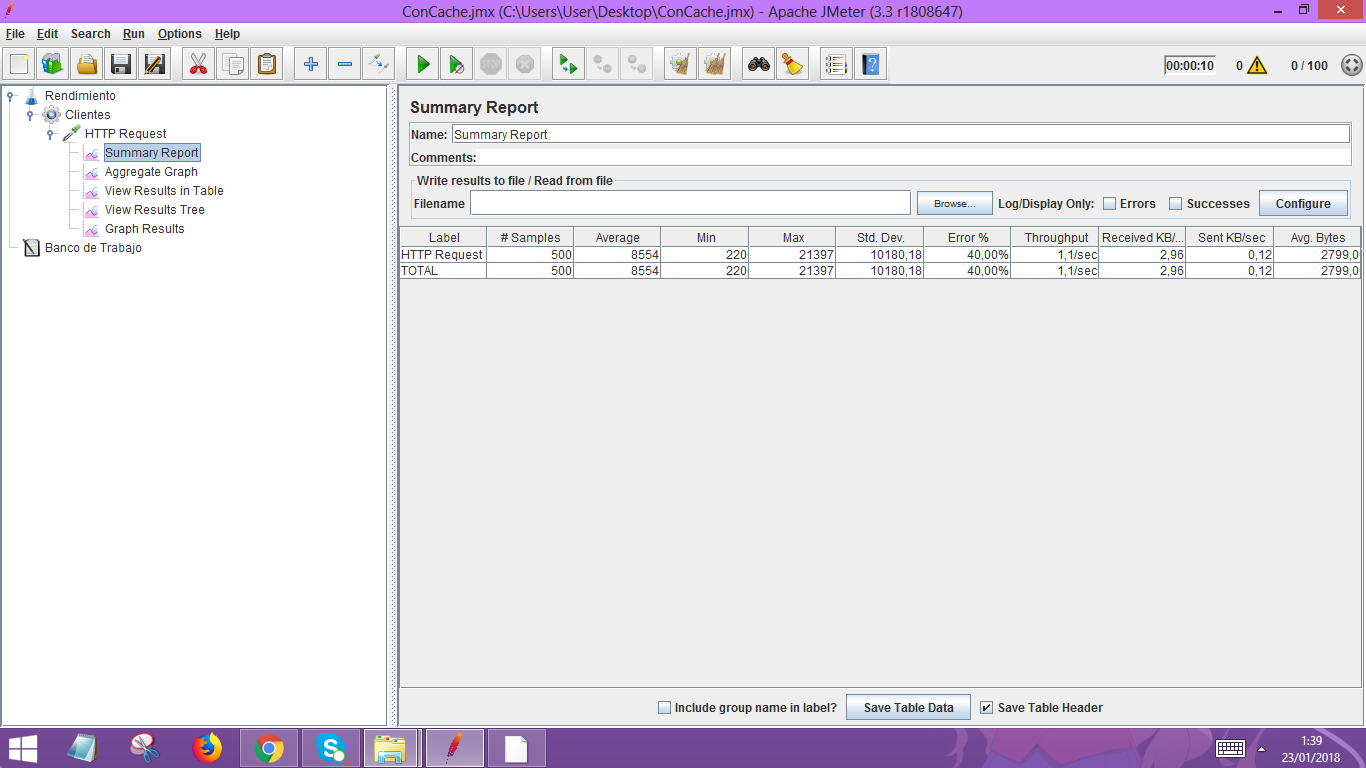


Ilustración . Resultados de la prueba de rendimiento B) presentados en un Summary Report (Se observa un valor de promedio de Througput de 1,1/seg )

Con los mismos parámetros del caso A , 100 clientes enviando pedidos en 10 segundos, podemos observar que las primeras solicitudes no son realizadas. Esto se debe a que, las noticias no se encontraban alojadas en el caché y se debía realizar en proceso de almacenamiento y lectura de la base de datos. Pero luego de realizar el “caching” los pedidos son ejecutados en un corto tiempo, obteniendo así un throughput de 1,1 solicitudes /seg