**PROYECTO DE SISTEMAS DISTRIBUIDOS**

Arquitectura de micro-servicios con caché para reducir latencia

Integrantes:

* Vicente Arauz
* Diego Robalino
* Rafael Suquinahua

Profesora:

* PhD. Cristina Abad

Paralelo: 1

**Documentación y justificación de decisiones de diseño del proyecto**

**Introducción**

El proyecto, está enfocado en implementar una aplicación web, con una arquitectura de microservicios y una caché para lograr reducir la latencia de acceso a la base de datos. El proyecto permitirá visualizar los diez gifs animados más populares del sistema. Los gifs, se encontrarán almacenado una base de datos. Para evitar saturar la base de datos, se implementó una caché en la cual se almacenará el resultado del query.

**Objetivos Generales**

* Implementar un sistema web con micro-servicios y una caché, para reducir la latencia.

**Objetivos Específicos**

* Desarrollar un sitio web, para presentar el top 10 de los gifts.
* Implementar el uso de la cache en el sistema.

**Estructuras de datos usados**

Para el proyecto se usó el lenguaje de Java donde implementamos las siguientes clases.

**BaseDatos:** La finalidad de la implementación de la clase BaseDatos es de ordenar la base de datos de gifts de mayor a menor según el número de visitas. Para después almacenarlos en una lista.

**Cache:** Implementamos el uso del memcached, para el almacenamiento de la caché, así reducir la latencia del sitio web.

**Comunicación:** Implementamos el uso del Reverse Proxy mediante el puerto 7911, esta clase permite la comunicación entre el Reverse Proxy y el servidor de microservicio.

**Convertidor:** La implementación de esta clase como su nombre lo menciona es de convertir el fichero proporcionado por la doctora de .tsv a un archivo .csv, además en esta clase se le agrega un número de visitas de valor aleatorio a las imágenes.

**Gift:** Declaramos en esta clase los atributos y métodos, que nos permite mostrar los datos almacenado y ordenados de la base datos.

**Microservicio:** Clase principal contiene el método top10

**Manejo de errores**

**Página web:** Se puso en producción el sitio web, al generar el top 10 mediante el botón se presenta los gifts, disminuyendo la latencia gracias al uso de la cache.

**Base de datos**: Se trabajo con la base de datos convirtiéndola de .tsv a .csv para evitar cualquier error que podría surgir adelante por la extensión que estaba.

**Lenguaje de programación usado.**

Se escogió Java por la experiencia que se tiene y la viabilidad de integrar con Apache thrift.

**Mecanismos de sincronización usados**

**Nginx Reverse Proxy**: Implementado para distribuir la carga, seguridad del sitio web.

**Memcached**: Implementado para el almacenamiento de la caché, así reducir la latencia del sitio web.

**MariaDB**: Implementada para la base de datos.

**Xampp**: Implementado para realizar las pruebas de forma local.

**Php**: Lenguaje implementado para hacer los request en el sitio web.

**Apache thrift**: Usado como middleware, el cual nos permite la conexión de cliente servidor, el cual el que actúa como cliente es nuestro sitio web y como servidor nuestro microservicio en java.

**Bootstrap**: Framework que utilizamos para trabajar el sitio web y personalizarlo a nuestro gusto.

**Css**: Implementado en el sitio web para mejorar la estética y agregarle diseños personalizados al sitio web.

**Aws**: Implementado para almacenar el sitio web y poder visualizarlo remotamente, siendo esta la nube usada.

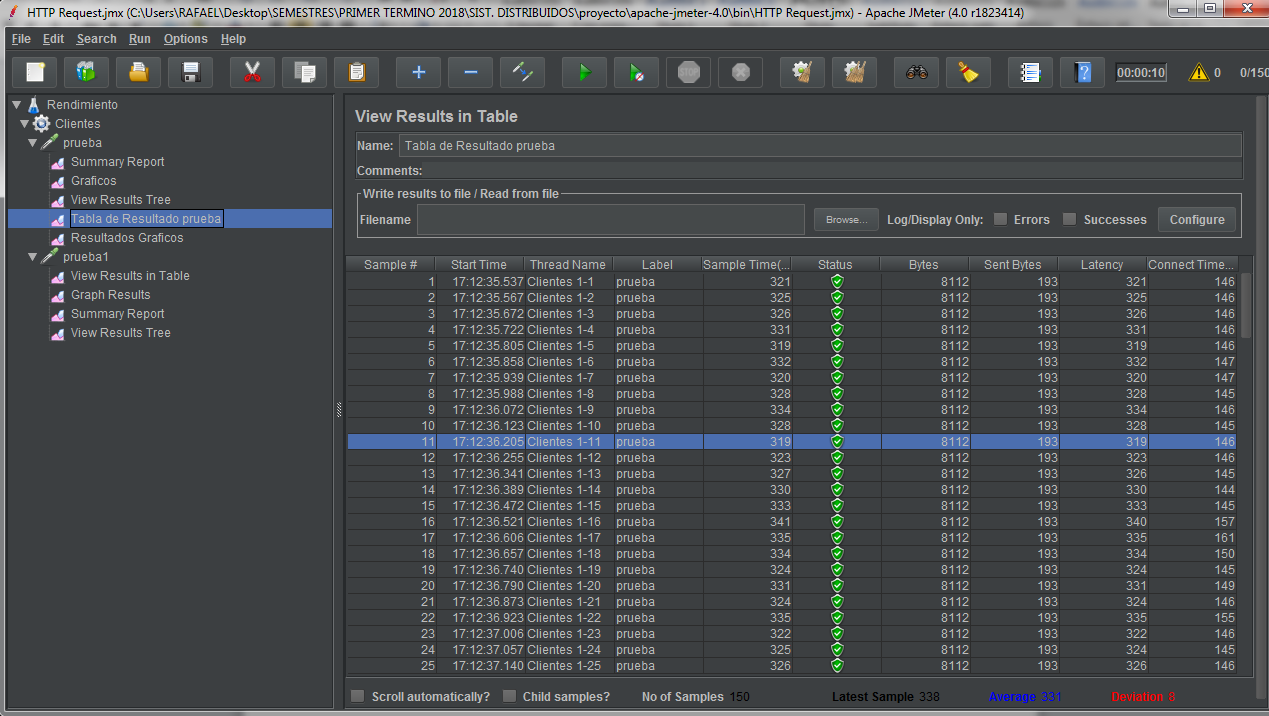


**Pruebas de rendimiento desde AWS**

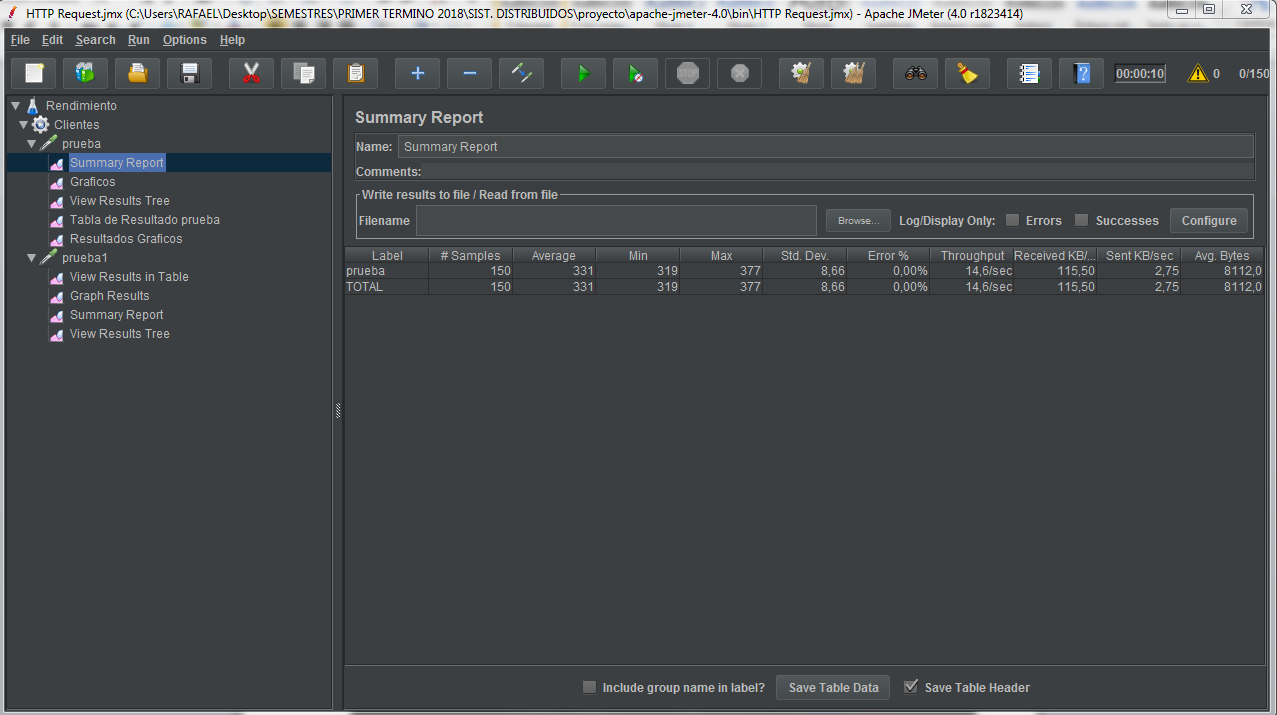
**APACHE JMETER**

Se usó Apache JMeter, para llevar a cabo de latencia y throughput es un software libre desarrollado en Java que permite evaluar y medir el desempeño de aplicaciones web enviando un número de peticiones al servidor en un intervalo de tiempo.

A continuación, se mostrarán los resultados que se realizaron en las pruebas en dos escenarios distintos:

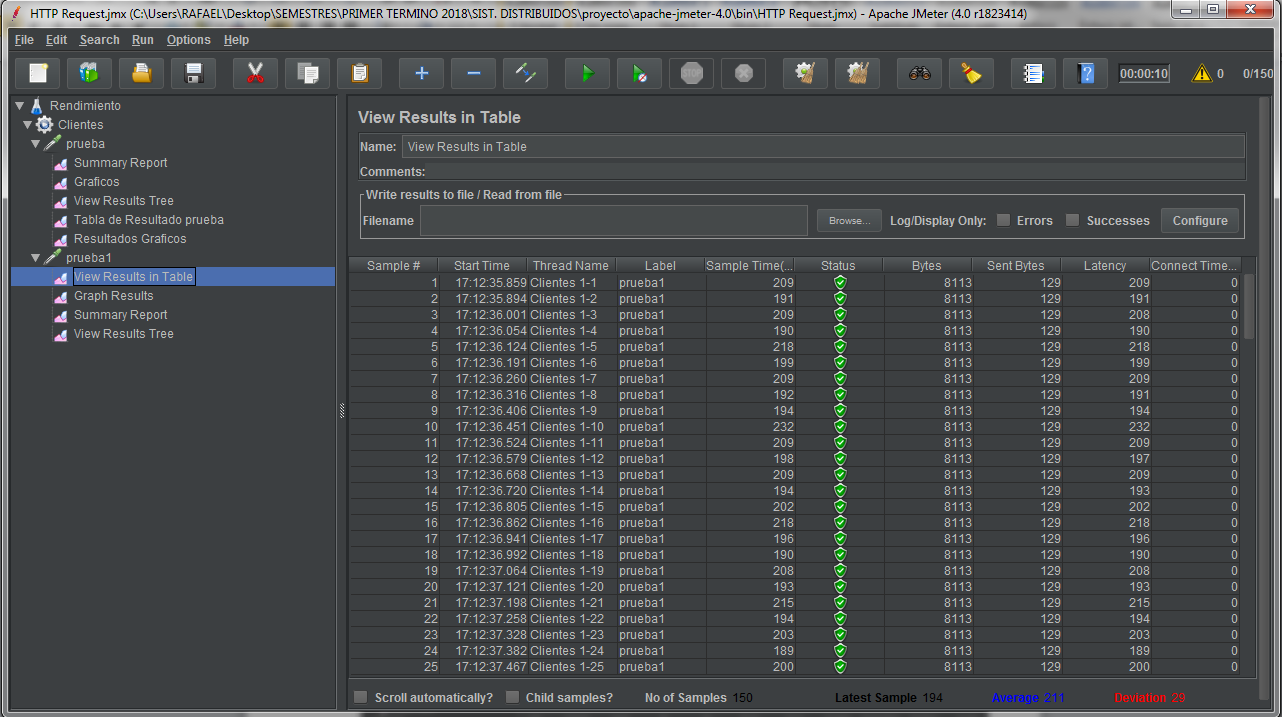
**MICROSERVICIO CON CACHE MAS BASE DE DATOS**

**Figura1: Resultados de la prueba1.php**



**Figura2:** Resultados de la prueba de rendimiento, presentados en un Summary Report (Se observa un valor de promedio de Througput de 14,6/seg )

Se puede observar con 150 clientes enviando pedidos en 10 segundos, las primeras solicitudes son realizadas en la figura1. Esto se debe a que, los gifs se encontraban alojadas en el caché. Pero luego de realizar el “caching” los pedidos son ejecutados en un corto tiempo, obteniendo así un throughput de 14,6 solicitudes /seg

**MICROSERVICIO USANDO SOLAMENTE LA BASE DE DATOS**

**Figura3. Resultados de la prueba1.php presentados en una tabla**

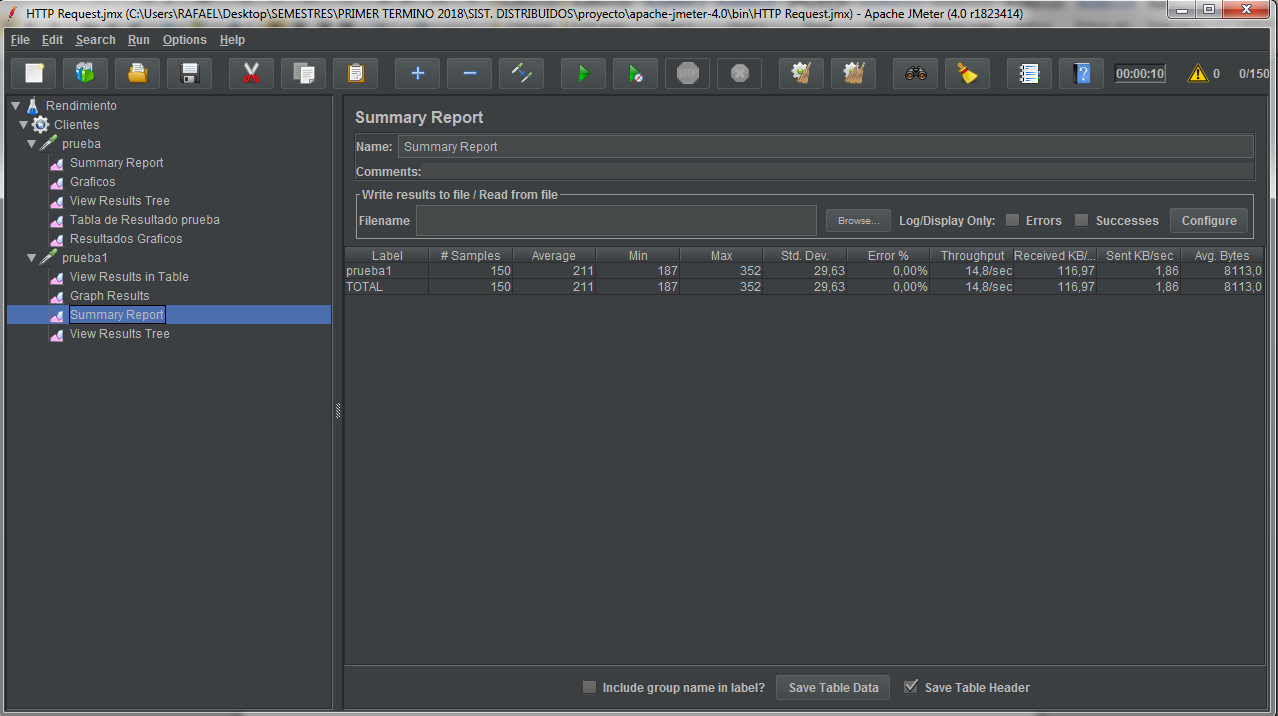


Ilustración 2. Resultados de la prueba de rendimiento, presentados en un Summary Report (Se observa un valor de promedio de Througput de 14,8/seg )

Se puede observar que con 150 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 14,8 solicitudes/seg.

**Cuartiles referente a la latencia**

**MICROSERVICIO CON CACHE MAS BASE DE DATOS**

|  |  |  |
| --- | --- | --- |
|  | **Valores** | **Anchos** |
| **MIN** | 319 | 319 |
| **Q1** | 325 | 6 |
| **Q2** | 330 | 5 |
| **Q3** | 336 | 6 |
| **MAX** | 377 | 41 |

|  |  |
| --- | --- |
| **RIC** | 11 |
| **Max** | 352,5 |
| **Min** | 308,5 |

Tabla1. Cuartiles de prueba.php

Tabla1 y tabla2. Muestra los datos de latencia caché más Base de Datos

**MICROSERVICIO USANDO SOLAMENTE LA BASE DE DATOS**

|  |  |  |
| --- | --- | --- |
|  | **Valores** | **Anchos** |
| **MIN** | 186 | 186 |
| **Q1** | 194 | 8 |
| **Q2** | 206,5 | 12,5 |
| **Q3** | 214,75 | 8,25 |
| **MAX** | 352 | 137,25 |

Tabla3. Cuartiles de prueba1.php

Tabla1 y tabla2. Muestra los datos de latencia solo con la Base de Datos