Proyecto 1, Entrega 4 - Arquitectura, conclusiones y consideraciones

Integrantes:

- Daniel Santamaría Álvarez - 201720222
- Nestor Gonzalez – 201912670
- Álvaro Plata – 201820098
- Rafael Humberto Rodriguez Rodriguez – 202214371

Los primeros cambios que se realizaron en la aplicación fueron para cambiar el manejo de las tareas asíncronas de la aplicación de Celery con redis a un esquema de publicador suscriptor con Cloud Pub/Sub.

En el servicio web primero se importa el SDK google cloud que contiene el módulo de Pub/Sub.

```
from google.cloud import pubsub_v1
```

Ahora, para configurar el servicio se debe tener el id del proyecto y el tópico al que se publicarán los mensajes, que en este caso es “tasks”. Luego se crea el cliente publicador con el destino de los mensajes.

```
topic_id="tasks"
project_id=os.environ.get("PROJECT_ID")
publisher=pubsub_v1.PublisherClient()
topic_path=publisher.topic_path(project_id,topic_id)
```

Para enviar un mensaje, por ejemplo en el endpoint de post tasks, los argumentos que se pasaban a celery se estructuran en un diccionario y se codifica su representación en string para poder ser enviado.

```
args={"task_id":task.id,"in_route":in_route,"out_route":out_route,"in_ext":in_ext,"out_ext":out_ext}
data=str(args).encode("utf-8")
```

Una vez está listo el mensaje de envío, se publica al tópico.

```
future=publisher.publish(topic_path,data)
```

Desde el worker, también se debe configurar el tópico al que se va a suscribir.

```
project_id = "swnube"
subscription_id = "tasks-sub"
```

```
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(project_id, subscription_id)
```

Por otro lado, se definió que se usaría un esquema de pull de los mensajes con exactly once delivery, para garantizar que solo 1 worker procese una tarea y que no se repita trabajo. En el stream de pull también se define una función de callback que se ejecutará cada vez que llegue un mensaje.

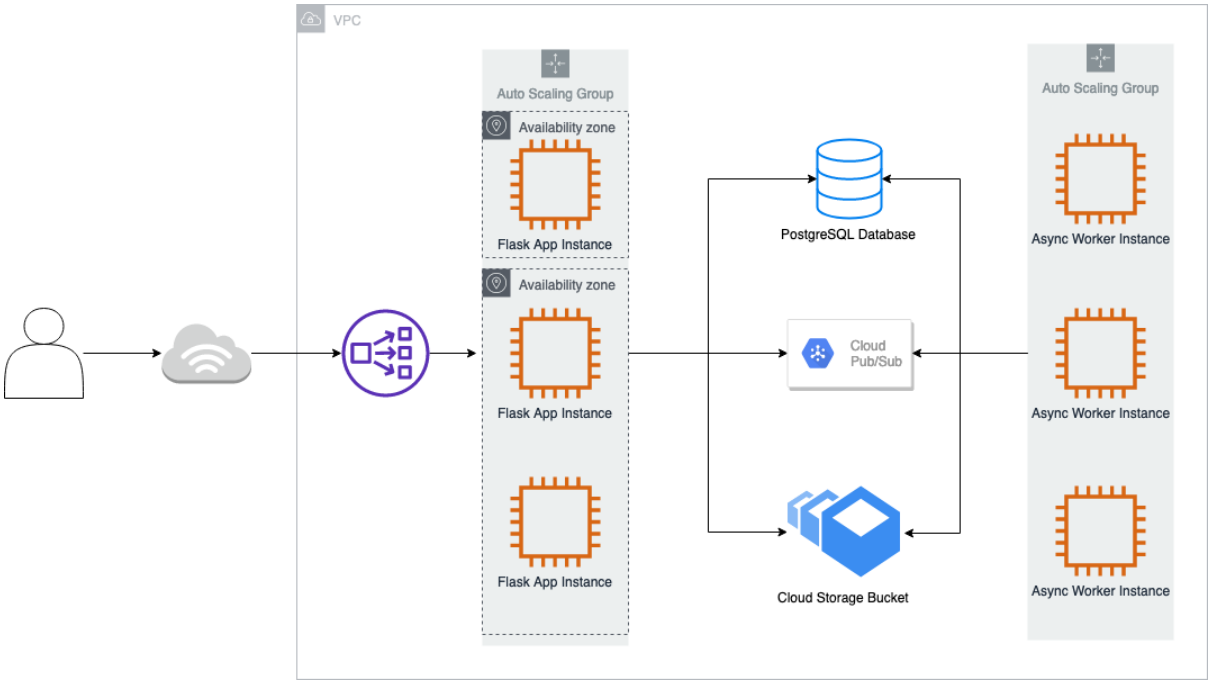
```
streaming_pull_future = subscriber.subscribe(subscription_path, callback=callback)
print(f"Listening for messages on {subscription_path}..\n")
```

La función de callback procesa el mensaje traído de la suscripción, lo decodifica y se usa la función eval que convierte el string a un diccionario. Después de esto, se pasan los parámetros a la misma función definida en entregas pasadas para hacer la conversión del archivo. Finalmente, el worker le notifica al servicio de Pub/Sub que recibió el mensaje exitosamente y sigue esperando a que llegue un mensaje nuevo.

```
def callback(message: pubsub_v1.subscriber.message.Message) -> None:
    print(f"Received {message}.")
    message_dict = eval(message.data.decode("utf-8"))
    print(message_dict)
    print(type(message_dict))
    start_conversion(message_dict["task_id"],message_dict["in_route"],message_dict["out_route"],message_dict["in_ext"],message_dict["out_ext"])

    # Use `ack_with_response()` instead of `ack()` to get a future that tracks
    # the result of the acknowledge call. When exactly-once delivery is enabled
    # on the subscription, the message is guaranteed to not be delivered again
    # if the ack future succeeds.
    ack_future = message.ack_with_response()
```

Descripción de la arquitectura desplegada en Google Cloud



A continuación, se describen los principales cambios realizados a la arquitectura de la aplicación desde la última entrega:

- Se reemplazó la aplicación de Celery por una aplicación en Python (Async Worker), que ejecuta el trabajo de convertir los formatos de los audios que se soliciten y guardarlos en Cloud Storage.

- Al no usar Celery, se creó un tópico del servicio Pub/Sub, donde se agregan las tareas cuando se realice una petición a la aplicación. La aplicación Web de Flask agrega las tareas al tópico y la nueva aplicación de Python (Async Worker) se suscribe y hace “pull” para verificar si hay nuevas tareas de forma periódica

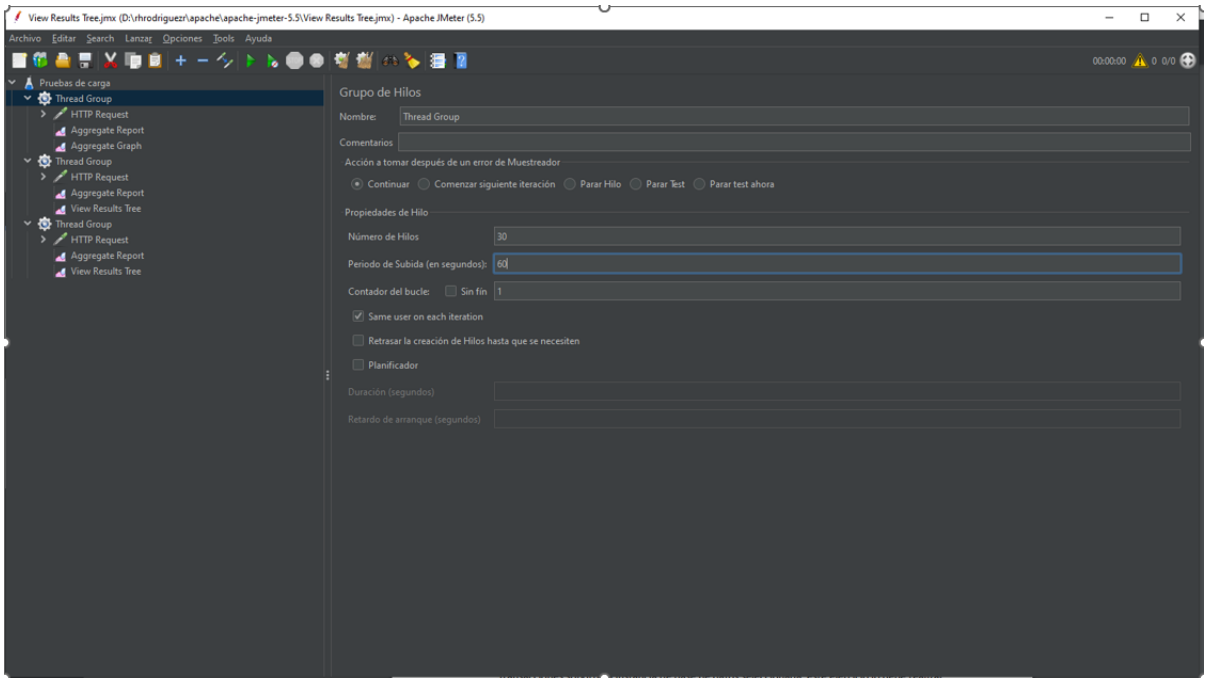
- Se configuró la aplicación Async Worker dentro de un grupo de auto escalamiento. De esta forma, se crearán o eliminarán instancias de Compute Engine con la aplicación dependiendo de la demanda.

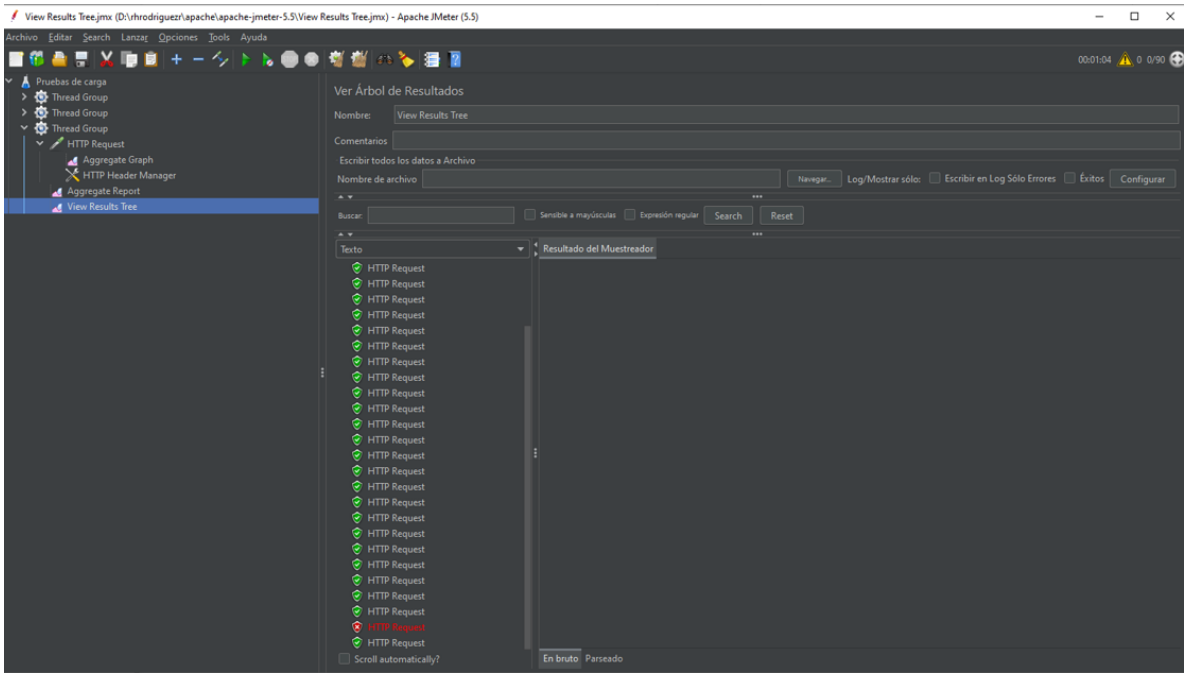
- Se configuró el grupo de auto escalamiento de la aplicación web de Flask para funcionar en 2 zonas de disponibilidad. De esta forma, al crearse instancias nuevas, estas podrán ser creadas en más de una zona, lo que aumenta los niveles de disponibilidad de nuestra aplicación y nos protege en caso de un desastre que elimine una zona de disponibilidad completa.

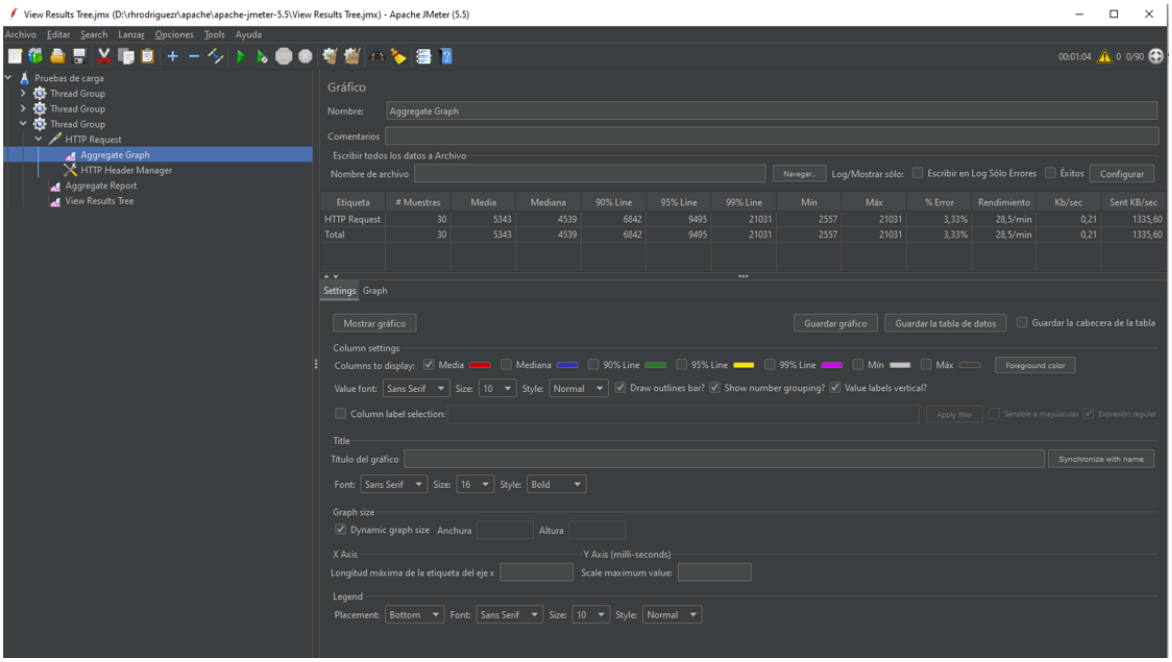
Análisis de Capacidad.

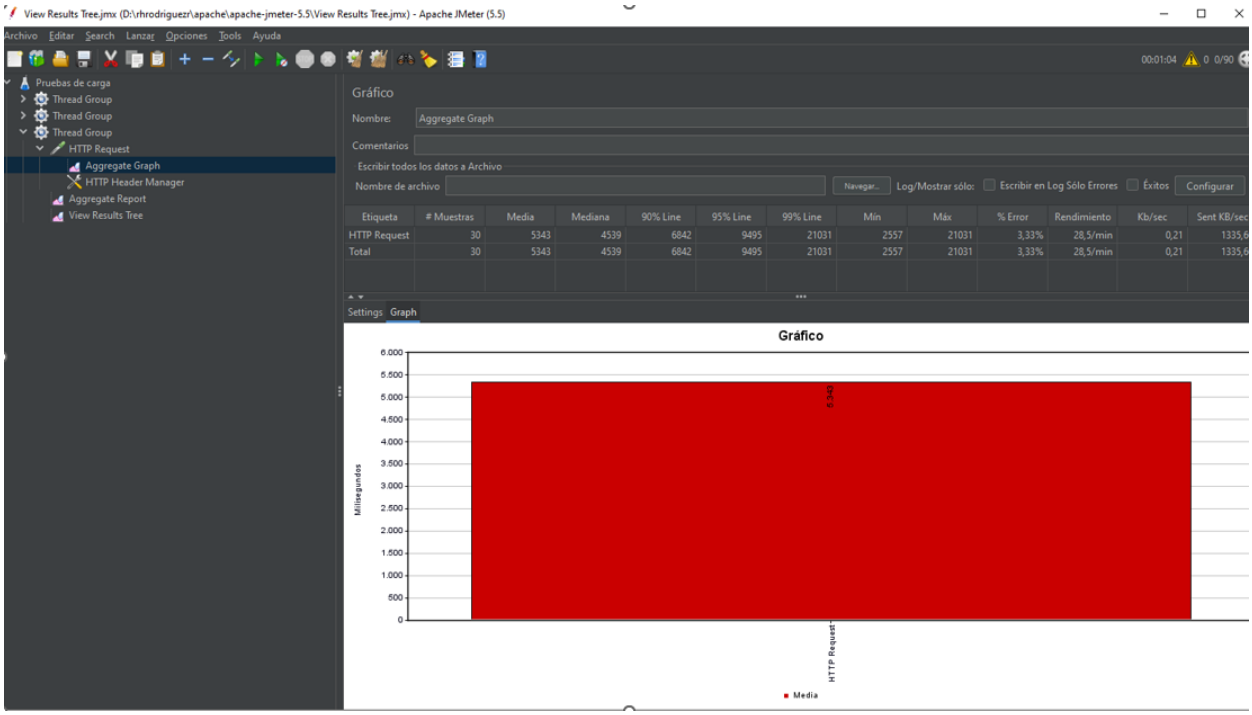
Escenario 1

Conversión de archivos



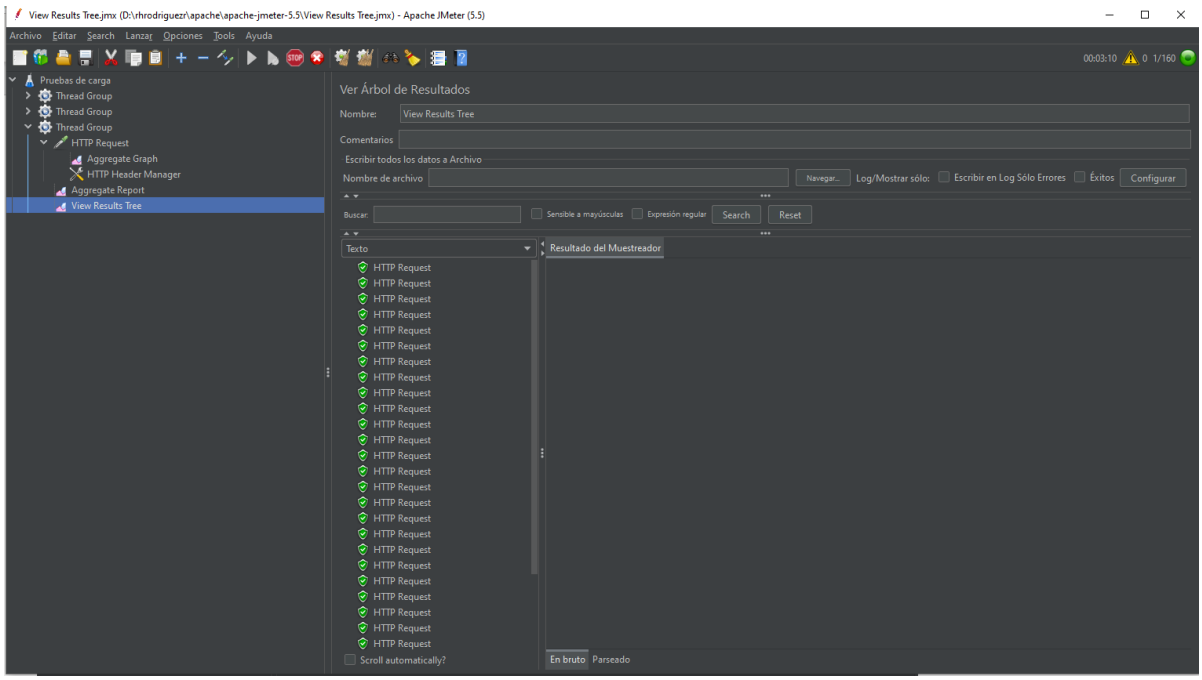

View Results Tree.jmx (D:\rhrodriguezr\apache\apache-jmeter-5.5\View Results Tree.jmx) - Apache JMeter (5.5)
Archivo Editar Search Lanzar Opciones Tools Ayuda
Pruebas de carga
Thread Group
Thread Group
Thread Group
HTTP Request
Aggregate Graph
HTTP Header Manager
Aggregate Report
View Results Tree
Ver Árbol de Resultados
Nombre: View Results Tree
Comentarios
Escribir todos los datos a Archivo
Nombre de archivo
Navegar...
Log/Mostrar sólo:
Escribir en Log Sólo Errores
Éxitos
Configurar
Buscar:
Sensible a mayúsculas
Expresión regular
Search
Reset
Texto
Resultado del Muestreador
HTTP Request
Scroll automatically?
En bruto
Parseado

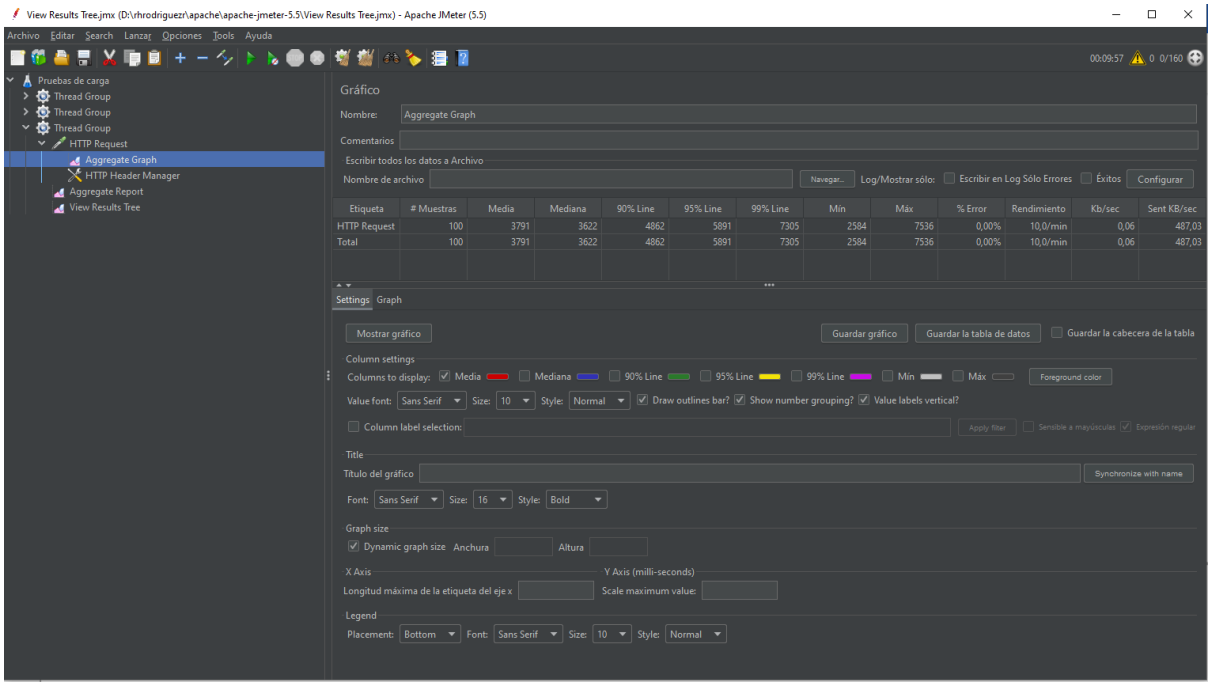

View Results Tree.jmx (D:\rhrodriguezr\apache\apache-jmeter-5.5\View Results Tree.jmx) - Apache JMeter (5.5)
Archivo Editar Search Lanzar Opciones Tools Ayuda
Pruebas de carga
Thread Group
Thread Group
Thread Group
HTTP Request
Aggregate Graph
HTTP Header Manager
Aggregate Report
View Results Tree
Gráfico
Nombre: Aggregate Graph
Comentarios
Escribir todos los datos a Archivo
Nombre de archivo
Navegar...
Log/Mostrar sólo:
Escribir en Log Sólo Errores
Éxitos
Configurar
Etiqueta # Muestras Media Mediana 90% Line 95% Line 99% Line Mín Máx % Error Rendimiento Kb/sec Sent KB/sec
HTTP Request 30 5343 4539 6842 9495 21031 2557 21031 3,33% 28,5/min 0,21 1335,60
Total 30 5343 4539 6842 9495 21031 2557 21031 3,33% 28,5/min 0,21 1335,60
Settings
Graph
Mostrar gráfico
Guardar gráfico
Guardar la tabla de datos
Guardar la cabecera de la tabla
Column settings
Columns to display:
Media
Mediana
90% Line
95% Line
99% Line
Mín
Máx
Foreground color
Value font:
Sans Serif
Size:
10
Style:
Normal
Draw outlines bar?
Show number grouping?
Value labels vertical?
Column label selection:
Apply filter
Sensible a mayúsculas
Expresión regular
Title
Título del gráfico
Synchronize with name
Font:
Sans Serif
Size:
16
Style:
Bold
Graph size
Dynamic graph size
Anchura
Altura
X Axis
Longitud máxima de la etiqueta del eje x
Y Axis (milli-seconds)
Scale maximum value:
Legend
Placement:
Bottom
Font:
Sans Serif
Size:
10
Style:
Normal

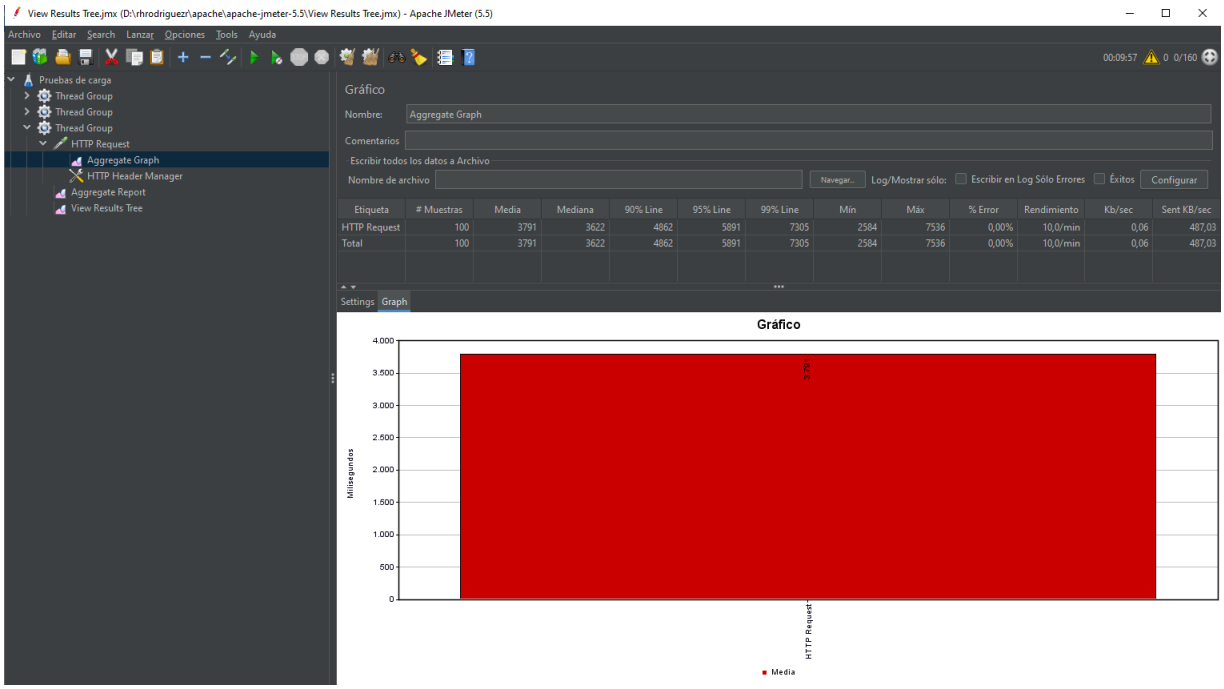


Realizando la comparación con la anterior entrega, mejoraron los tiempos de respuesta, disminuyeron los porcentajes de error y se obtiene un mejor rendimiento.

Escenario 2







Respecto a la anterior entrega el porcentaje bajo considerablemente, el promedio de ejecución subió y se mantiene el rendimiento de conversión de 10 archivos por minuto.