

Proyecto 1, Entrega 3 - Arquitectura, conclusiones y consideraciones

Integrantes:

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

Para poder incorporar los cambios de la infraestructura, primero incorporamos el SDK de google cloud, específicamente el módulo de almacenamiento.

```
from google.cloud import storage
```

Después, se debe obtener el cliente, que corresponde a la cuenta de servicio de la maquina virtual, para obtener acceso al API de los buckets de storage. Para poder acceder al bucket se debe usar su nombre único definido en la creación.

```
storage_client = storage.Client()
bucket = storage_client.bucket(bucket_name)
```

Con la conexión al bucket, se deben usar blobs que obtienen y almacenan los archivos de audio en su correspondiente bucket. Por ejemplo, en la estructura definida para el almacenamiento del proyecto, habrá una carpeta por conversión con el id asignado a la tarea y en su interior contiene el archivo original y el modificado. Para cargar el archivo se usa la función `upload_from_file` del blob.

```
in_route = "{}/{}/".format(task.id,filename)
out_route = "{}/{}/".format(task.id,filename.replace(f'.{in_ext.lower()}',f'.{out_ext.lower()}'))
blob = bucket.blob("{}{}{}".format(task.id, filename))
blob.upload_from_file(original_audio)
```

Igualmente, en el procesamiento asíncrono, se obtiene el archivo original mediante la variable `in_route` como ruta del blob y se descarga el archivo a una ruta local. A partir de este archivo, se hace la conversión y en un blob con la ruta de destino se carga el nuevo archivo.


```
blob = bucket.blob(in_route)
blob.download_to_filename("{}{}{}".format(in_route.split("/")[1]))
AudioSegment.from_file("{}{}{}".format(in_route.split("/")[1],format=in_ext)).export("{}{}{}".format(out_route.split("/")[1], format=out_ext))
blob = bucket.blob(out_route)
blob.upload_from_filename(out_route.split("/")[1])
```

Para poder controlar la petición que realiza el procesamiento asíncrono al finalizar una tarea, se definió la variable de entorno `API_INSTANCE_IP` que apunta a la dirección del balanceador de carga para que redirija la petición a un servidor web disponible.


```
r = requests.post(f'http://{os.environ.get("API_INSTANCE_IP")}:5000/api/tasks/{task_id}/processed')
```

De acuerdo con los cambios anteriores, se puede ver el bucket con la estructura definida:

swnube30




Ubicación	Clase de almacenamiento	Acceso público	Protección
us-east1 (Carolina del Sur)	Standard	 Público para Internet	Ninguna

OBJETOS CONFIGURACIÓN PERMISOS PROTECCIÓN CICLO DE VIDA OBSERVABILIDAD

Depósitos > swnube30 > 41 

[SUBIR ARCHIVOS](#) [SUBIR CARPETA](#) [CREAR CARPETA](#) [TRANSFERIR LOS DATOS](#) [ADMINISTRAR C](#)
[BORRAR](#)

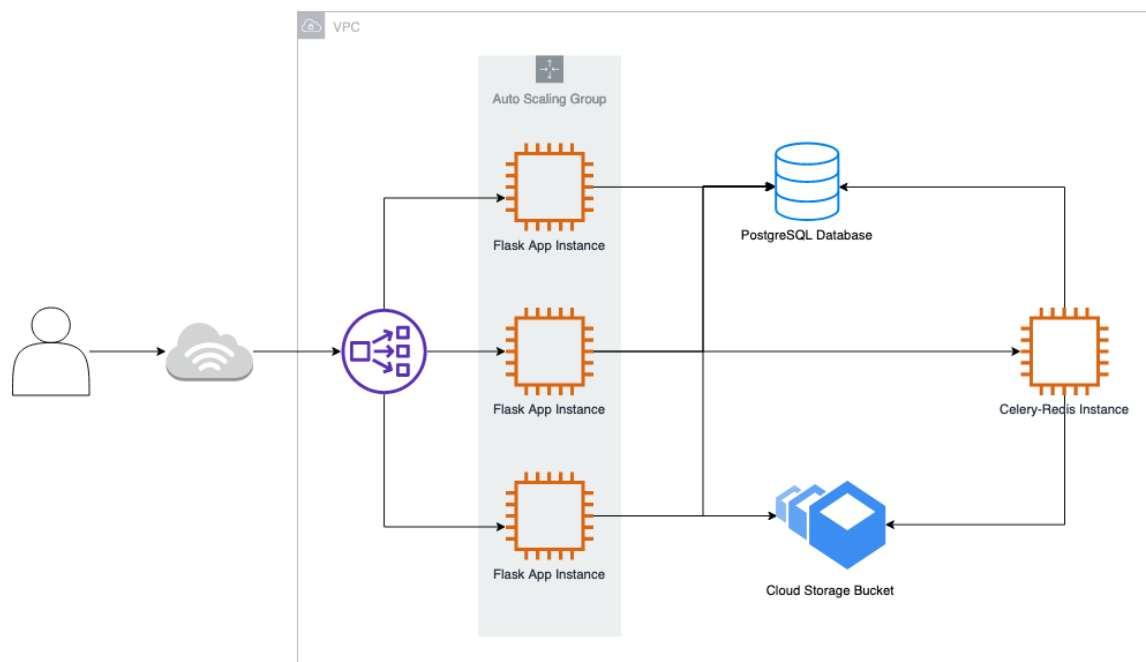
Filtrar solo por prefijo de nombre  **Filtro** Filtrar objetos y carpetas

<input type="checkbox"/>	Nombre	Tamaño	Tipo	Fecha de creación 
<input type="checkbox"/>	 VuelveLatinDreams.mp3	2.8 MB	application/octet-stream	13 nov 2022 23:59:...
<input type="checkbox"/>	 VuelveLatinDreams.wav	31.3 MB	audio/x-wav	13 nov 2022 23:59:...

Para poder crear instancias con escalamiento automático, se definió un script que se ejecuta siempre que se crea una instancia para instalar las dependencias y el código que se despliega:

```
#!/bin/bash
sudo apt-get update
sudo apt-get install python3
sudo apt-get install python3-pip -y
sudo apt-get install git -y
sudo apt-get install ffmpeg -y
sudo pip3 install virtualenv
git clone https://github.com/NestorGon/SWNube.git
cd SWNube
virtualenv venv
source bin/activate
pip3 install -r requirements.txt
cd flaskr
export REDIS_INSTANCE_IP=35.238.0.106
python3 -m flask run --host=0.0.0.0
```

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:

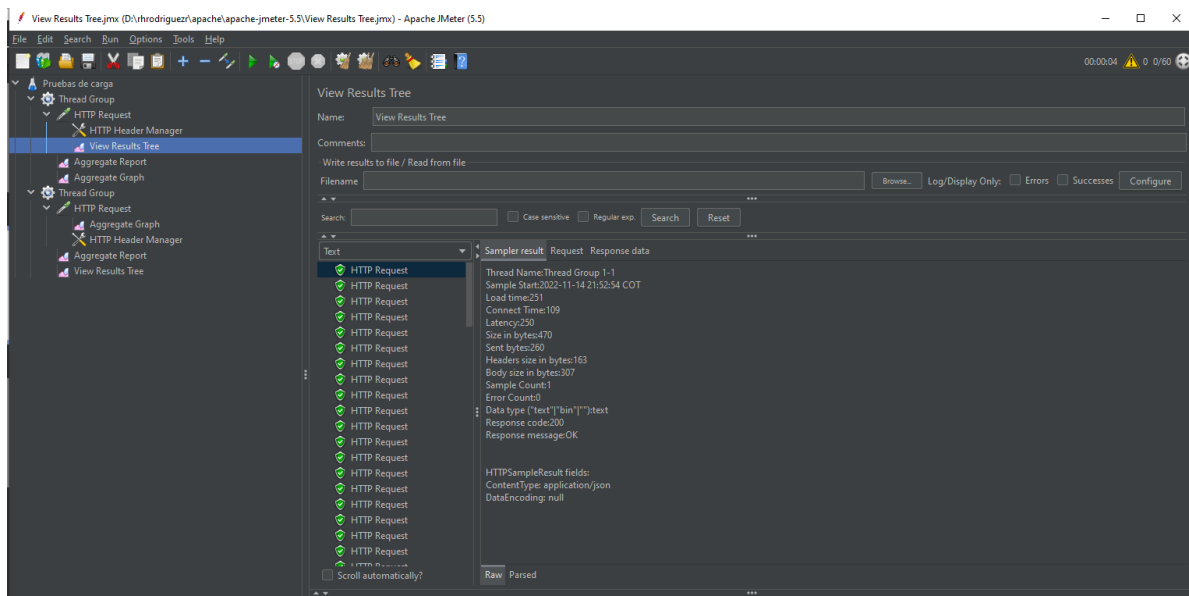
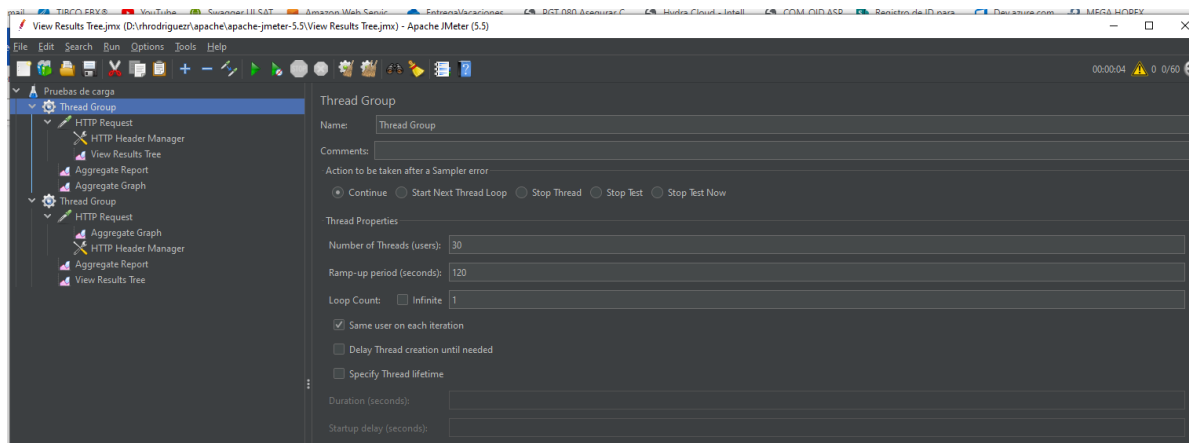
- Sustitución del servidor de NFS por el uso de un servicio administrado de almacenamiento de Objetos, Google Cloud Storage, para el almacenamiento de los archivos de audio originales y compartidos.
- Creación de una Plantilla de Instancias y un Grupo de Instancias de auto escalamiento. Estos elementos reemplazaron a la instancia única que anteriormente se encontraba sirviendo la aplicación de Flask (capa web). De esta forma, ahora la capa web puede escalar y desescalar automáticamente dependiendo del tráfico que reciba la aplicación.
- Creación de un Balanceador de Carga, que distribuye la carga y apunta hacia el grupo de instancias que se crean y eliminan de manera dinámica, de manera que las peticiones que vienen directamente de los usuarios ahora apuntan al balanceador en lugar de una instancia específica.

Análisis de Capacidad.

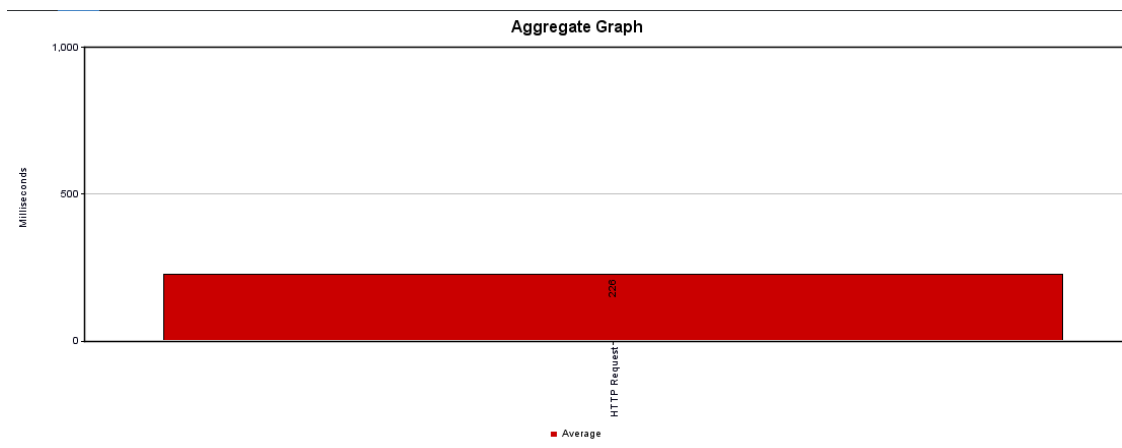
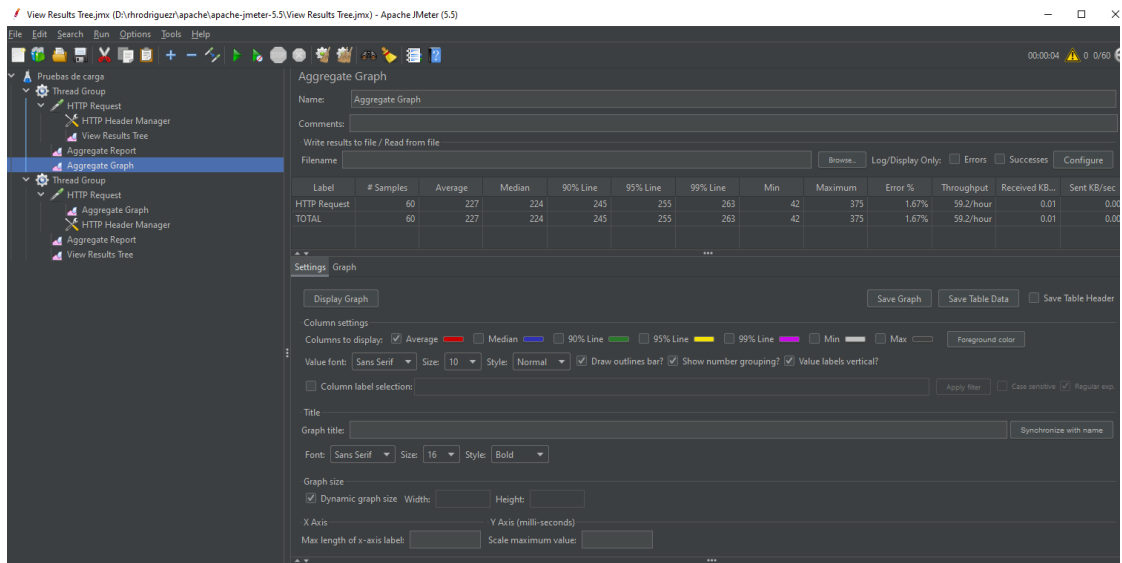
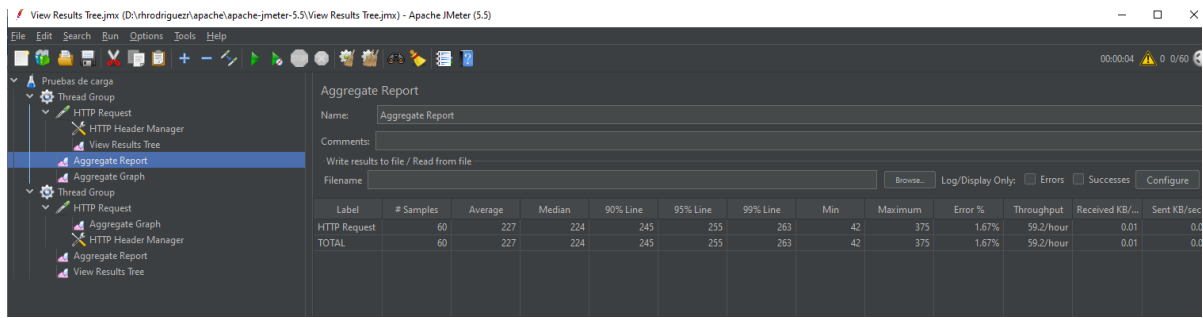
Pruebas de estrés sobre el Modelo de Despliegue

Escenario 1

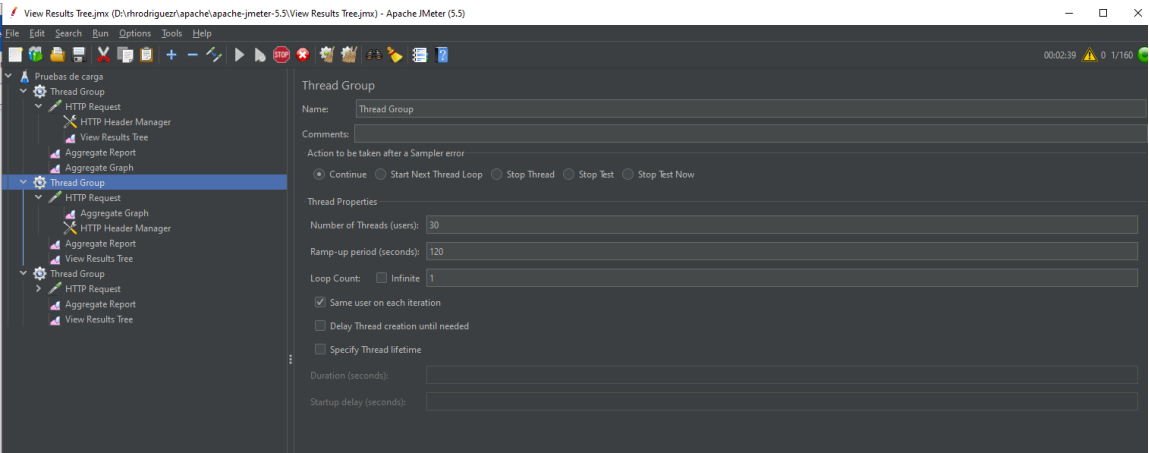
Login



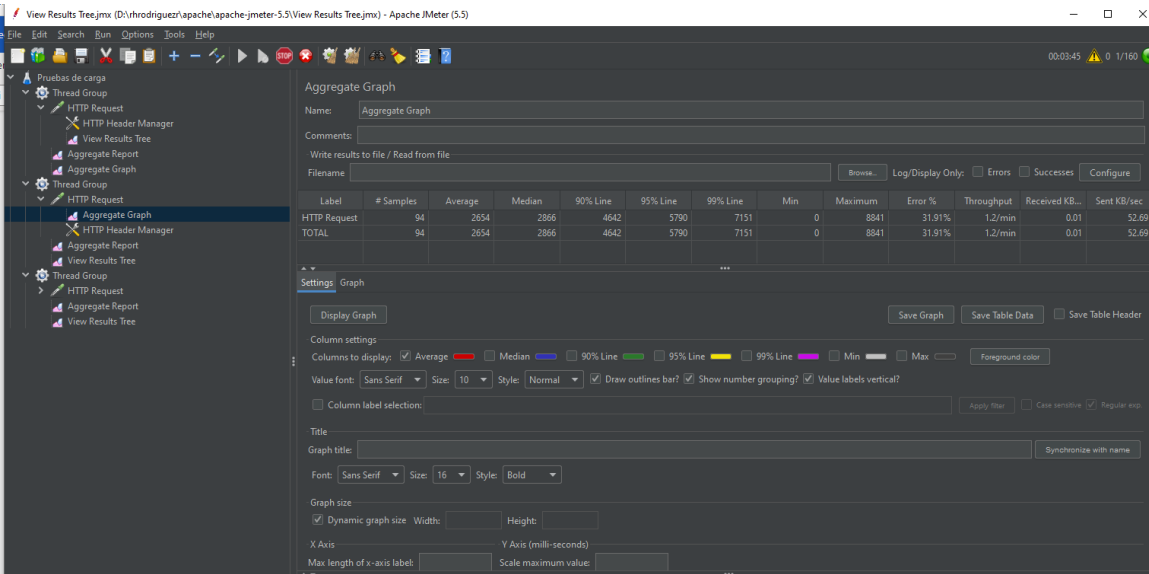
El porcentaje de error supera el 1 % por lo cual se puede deducir que la prueba es fallida.



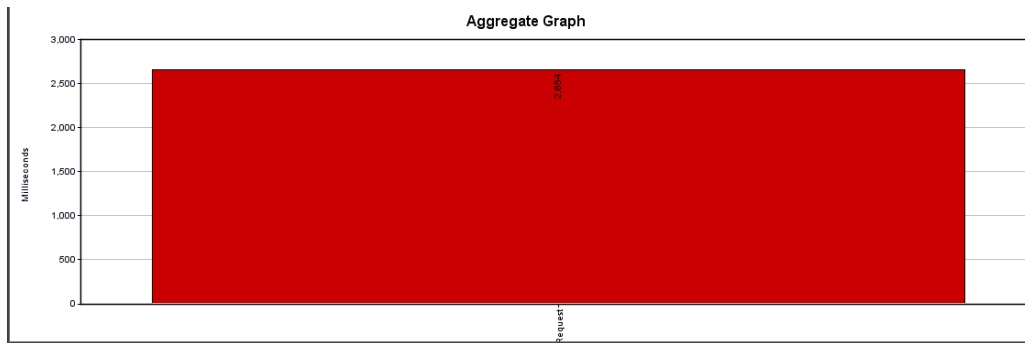
Conversión de archivos.



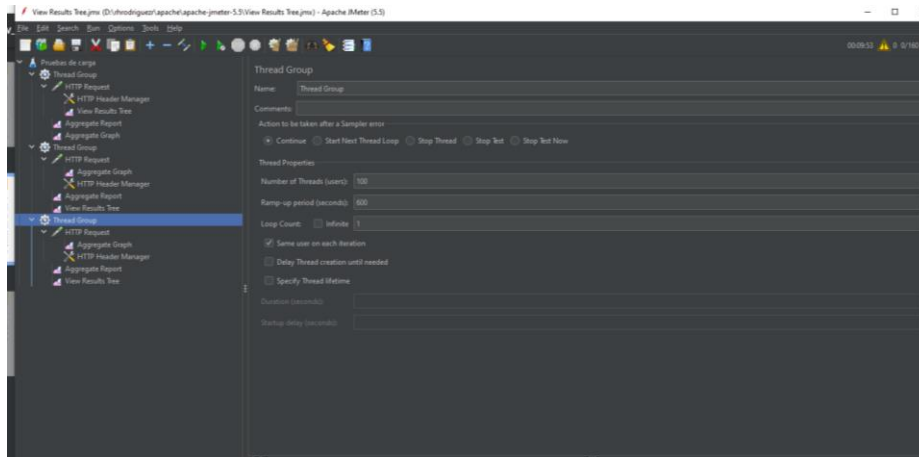
El porcentaje de error es bastante alto por lo cual también se puede dar por fallida la prueba.



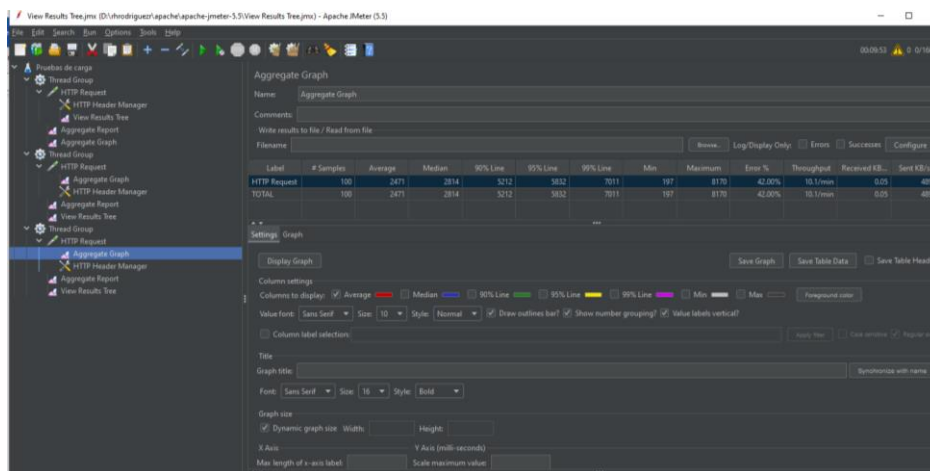
Se supera el tope propuesto del escenario de 1.5 ms por petición



Escenario 2



El porcentaje de conversiones por minuto es de 10 archivos.



View Results Tree.jmx (D:\thorodriguez\apache\apache-jmeter-5.5\View Results Tree.jmx) - Apache JMeter (5.5)

File Edit Search Run Options Tools Help

00:09:53 0/160

Pruebas de carga

- Thread Group
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - Aggregate Report
 - Aggregate Graph
 - Thread Group
 - HTTP Request
 - Aggregate Graph
 - HTTP Header Manager
 - View Results Tree
 - Aggregate Report
 - Aggregate Graph
 - Thread Group
 - HTTP Request
 - Aggregate Graph
 - HTTP Header Manager
 - View Results Tree
 - Aggregate Report
 - Aggregate Graph

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes ☐ Configure

Search: Case sensitive ☐ Regular exp. ☐ Search

Text

Sampler result	Request	Response data
HTTP Request	Thread Name: Thread Group 3-59	
HTTP Request	Sample Start: 2022-11-14 23:31:15 COT	
HTTP Request	Load time: 226	
HTTP Request	Connect Time: 105	
HTTP Request	Latency: 229	
HTTP Request	Size in bytes: 237	
HTTP Request	Sent bytes: 2981341	
HTTP Request	Headers size in bytes: 200	
HTTP Request	Body size in bytes: 27	
HTTP Request	Sample Count: 1	
HTTP Request	Error Count: 1	
HTTP Request	Data type: ("Test")text	
HTTP Request	Response code: 500	
HTTP Request	Response message: Internal Server Error	
HTTP Request	HTTPSampleResult fields	
HTTP Request	ContentType: application/json	
HTTP Request	DataEncoding: null	

Raw Parsed

Scroll automatically?

```
2022-11-14 23:31:15,315 INFO o.a.j.m.h.s.HTTPSampleResult - Sample result on the requests will be replaced by a HTTPSampleResult - Max request is: 1
997 2022-11-14 23:35:21,712 INFO o.a.j.t.JMeterThread: Thread is done: Thread Group 3-100
998 2022-11-14 23:35:21,712 INFO o.a.j.t.JMeterThread: Thread finished: Thread Group 3-100
999 2022-11-14 23:35:21,713 INFO o.a.j.e.s.StandardJMeterEngine: Notifying test listeners of end of test
1000 2022-11-14 23:35:21,713 INFO o.a.j.g.u.JMeterReminders: setRunning(false, "local")
1001
```

Aggregate Graph

