Los atributos de calidad que serán evaluados en este experimento son:

- Disponibilidad: Se espera que el 99.95% de las solicitudes de reserva de los usuarios de Mobibus y consultas de los operadores de Tbc sean atendidas. Además ante la falla del servidor principal de Tbc, otra instancia pueda asegurar la operación del sistema.
- Modificabilidad: Determinar y mejorar la mantenibilidad del desarrollo logrado con el proyecto. Se debe establecer con Sonar las métricas: complejidad, duplicaciones, líneas de código e issues. Igualmente se debe definir y aplicar un conjunto de lineamiento de codificación para lograr un código más mantenible en el proyecto.

Pre-Experimentación:

1. Problemática: El problema principal expuesto en el enunciado es la necesidad de poder garantizar que el sistema de TBc cumpla con atender el 99.95% de las solicitudes y el de determinar y mejorar la mantenibilidad del desarrollo logrado hasta ahora.
2. Objetivo: Se plantea utilizar un DNS que apunta a ambos servidores que se apuntan entre sí como doble balanceador de carga, de esta forma la disponibilidad del sistema estará garantizada. Adicional a esto se utilizara sonar para garantizar la modificabilidad.
3. Descripción del experimento: Las actividades a realizar son:

➔ Realizar modificaciones a la base de datos necesarios para la realización del experimento 3.

➔ Diseñar los artefactos necesarios para garantizar el atributo de disponibilidad.

➔ Diseñar los artefactos necesarios para garantizar el atributo de modificabilidad utilizando sonar PhP.

➔ Desarrollar las pruebas de carga con loader.io.

➔ Realizar la documentación del experimento 3, entrega parcial.

Y los datos a recolectar son:

➔ Datos de disponibilidad: ver el porcentaje de las solicitudes que se atienden.

4. Los artefactos a construir inicialmente son: No se construirá uno nuevo, sino que se modificara y mejorará los que ya se hicieron para el experimento 1 y se implementará un DNS que apunta a ambos servidores para el balanceador de carga.

5. Los recursos que se van a utilizar para el experimento son:

- El IDE PhpStorm: para el desarrollo del código.
- Php lumen: para la escritura del código.
- Github: para un repositorio común de todos los miembros.
- loader.io para la realización de pruebas del codigo y ruteo.
- MariaDB: para realizar la capa de persistencia.
- Sonar PhP para la modificabilidad.

6. Los resultados esperados son:

- Que el sistema atienda el 99.95% de las solicitudes.
- Que los resultados obtenidos en el experimento 1 no se vean afectados.

7. El tiempo esperado, tomando en cuenta que la disponibilidad se realizo en el exp 2, entonces únicamente se espera 2 horas para el atributo de modificabilidad.

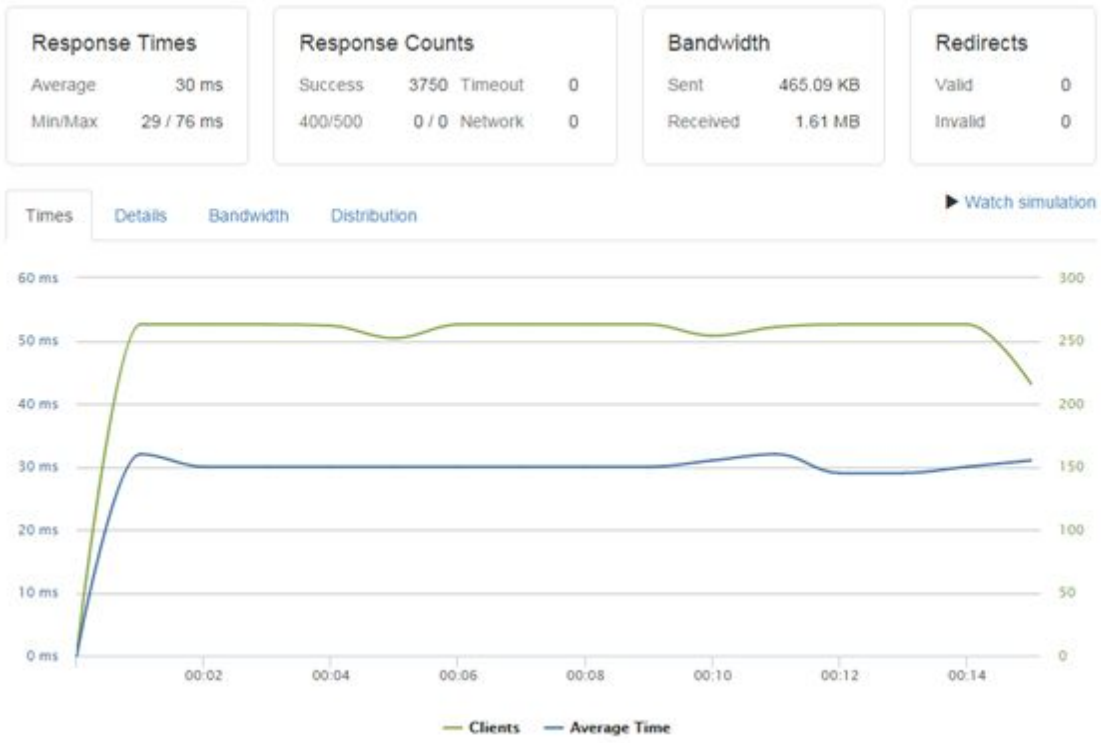
1. Resultados Obtenidos

Los casos de uso identificados en la primera entrega son los mismos que para esta entrega.

Pruebas de latencia - Casos de uso:

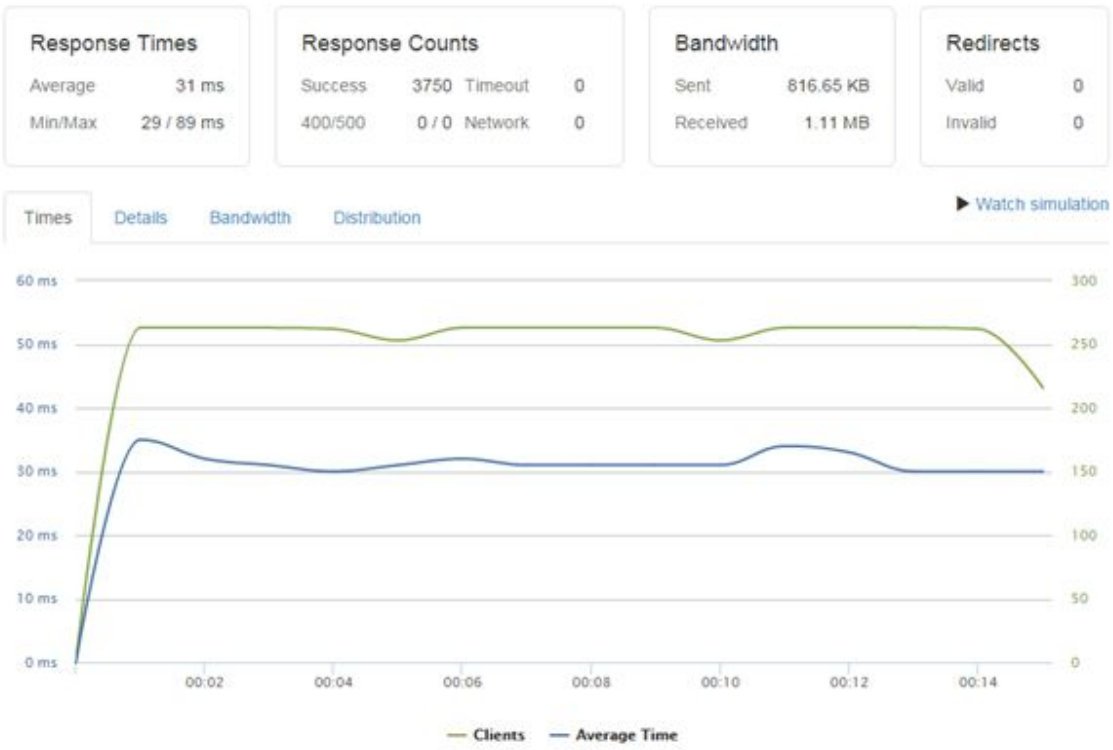
CASOS PARA EL MOBIBUS:

1. Ver un mobibus (GET):



2. Reportar la posición de un mobibus (POST):

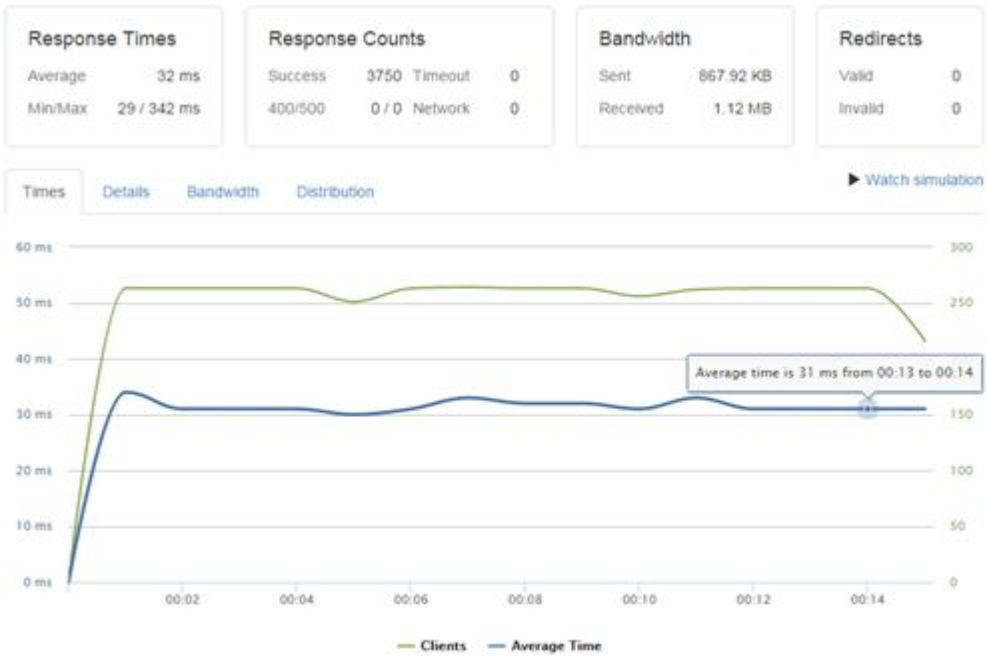
3. Ocupar un Mobibus (POST)



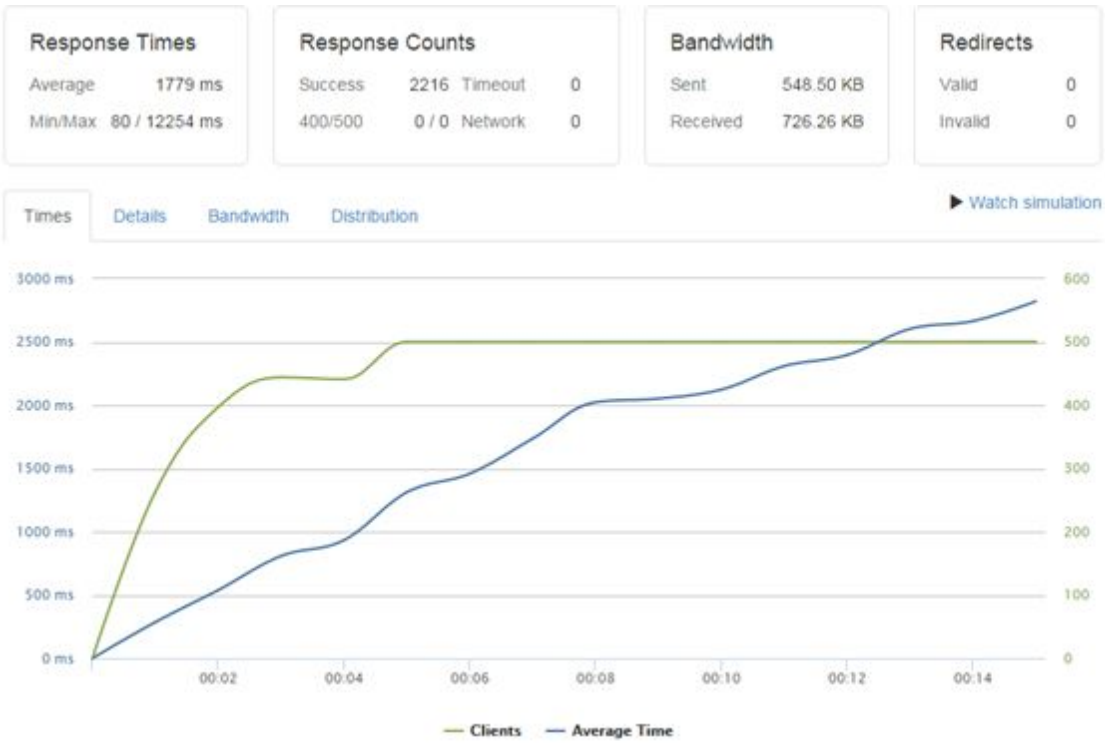
CASOS PARA TRANVÍA:

1. Obtener un tranvía (GET):

2. Reportar La posición (POST)

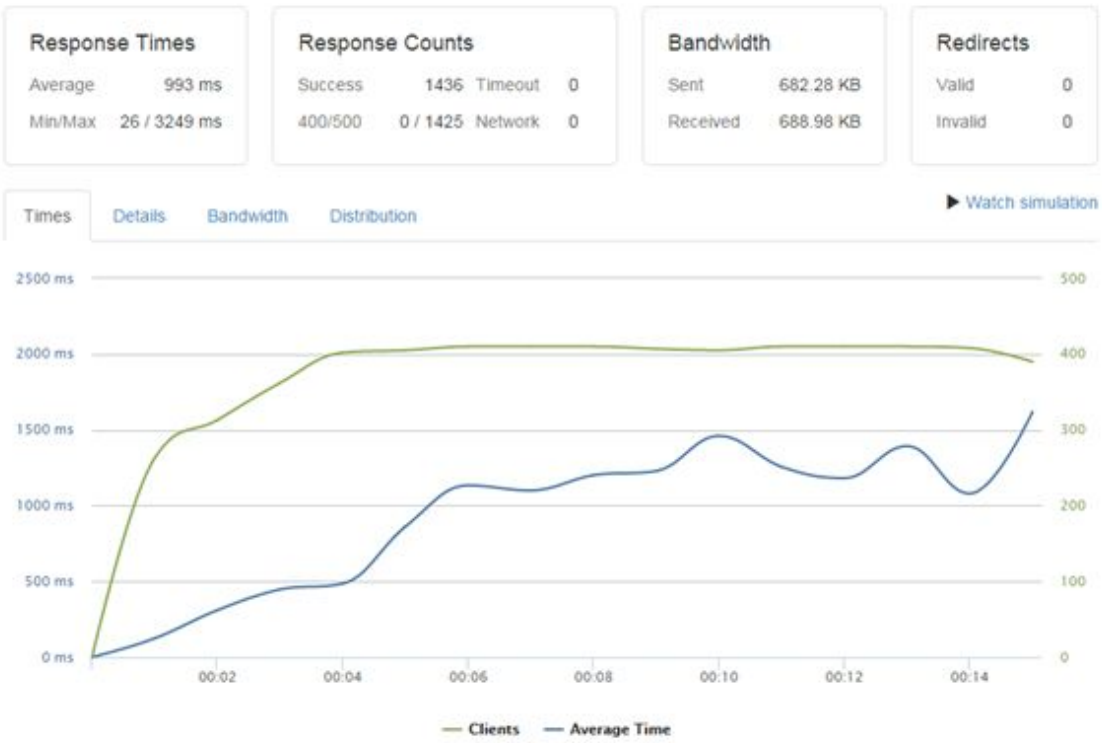


3. Reportar Emergencia



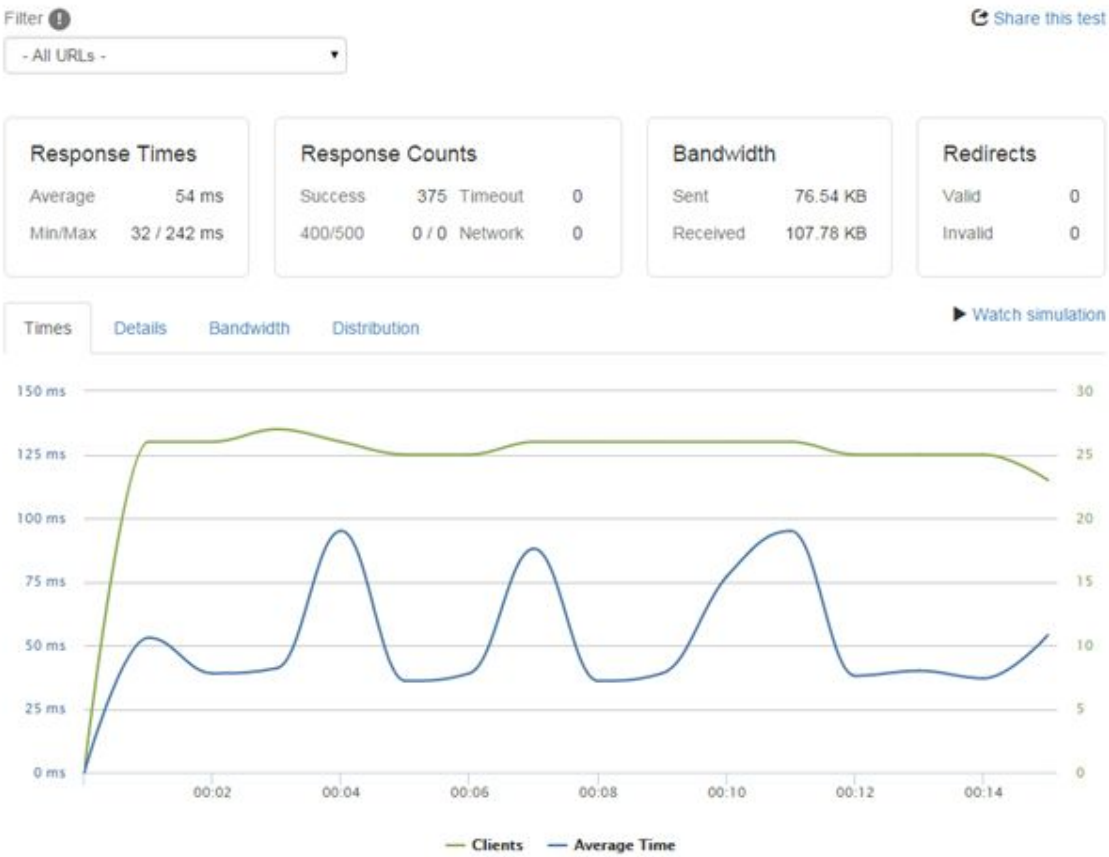
CASOS PARA VCUBS:

1. Registrar un Vcub en una terminal (POST):

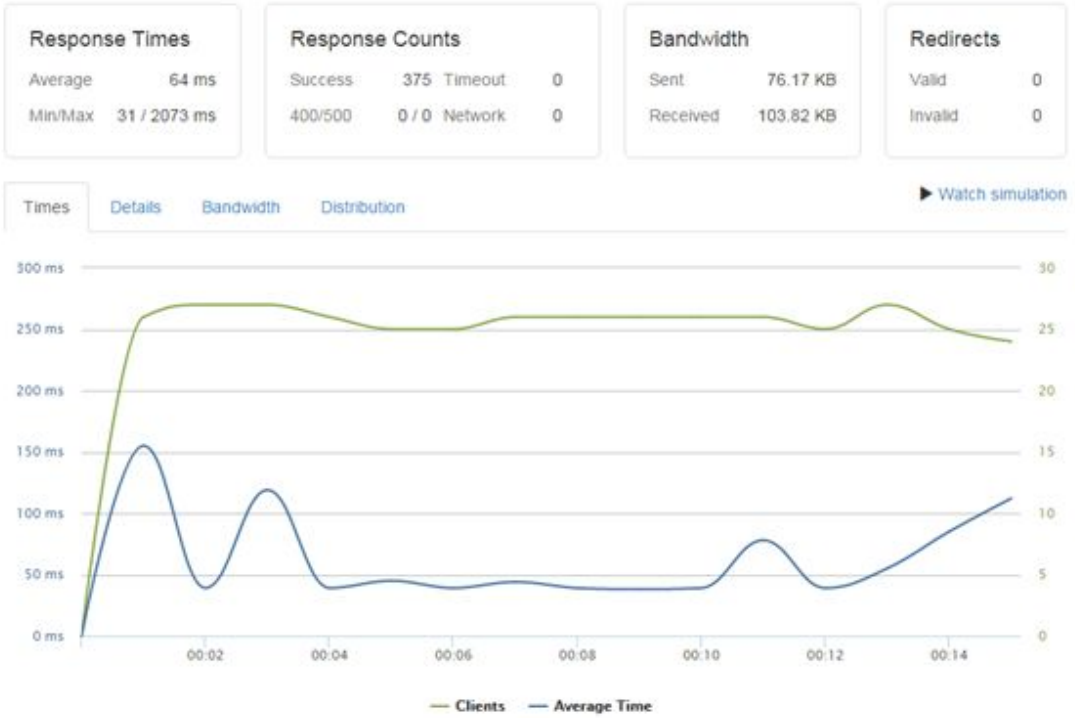


2. Prestar un Vcub (POST):

3. Recibir un Vcub (PUT):

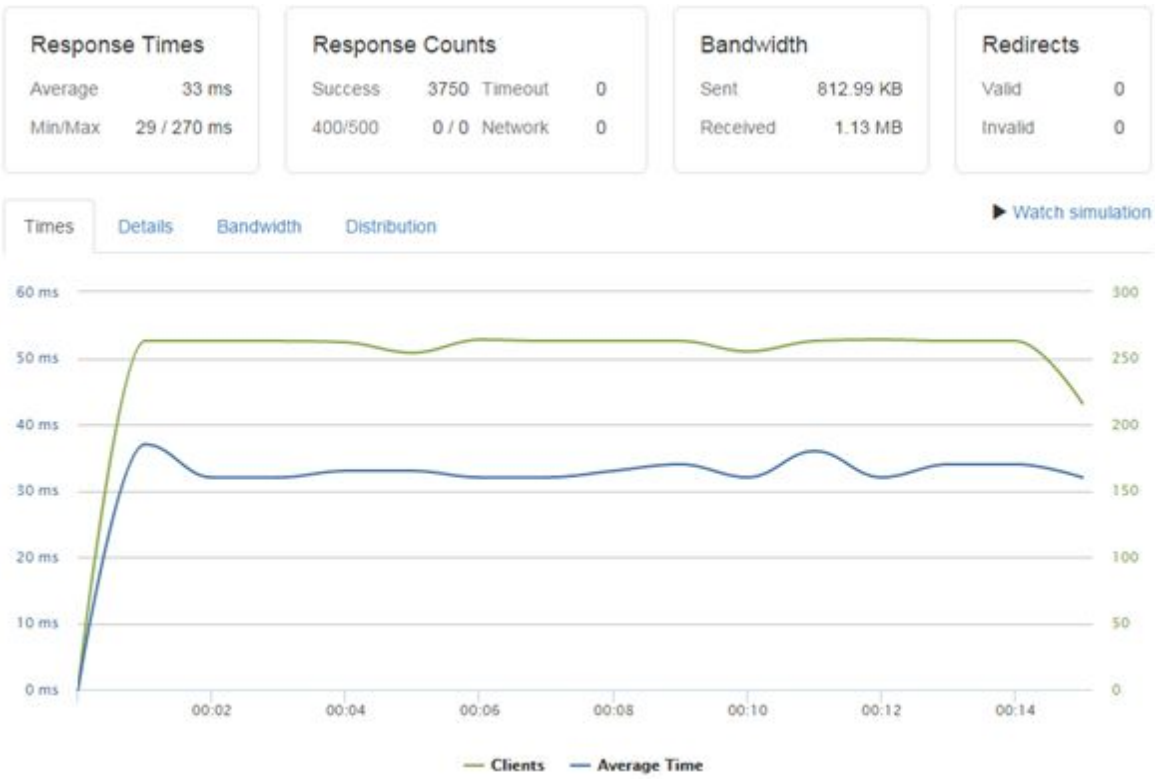


4. Pedir un llenado de Vcubs:

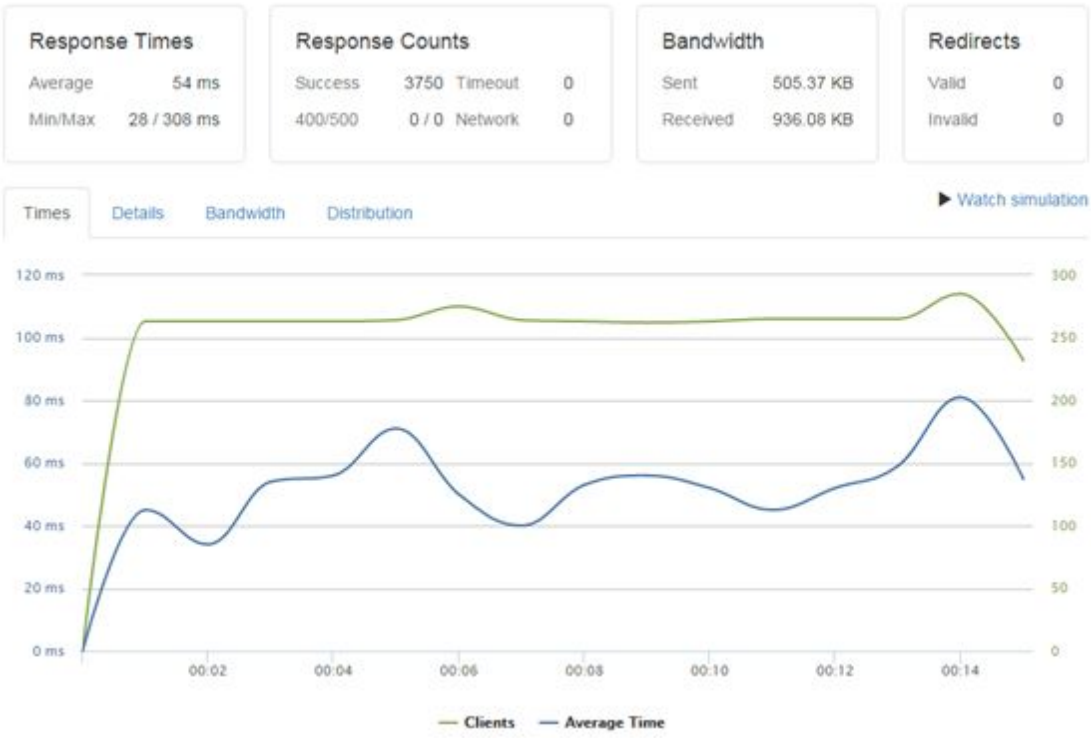


CASOS PARA Usuario:

1. **Solicitar un mobibus (GET):**



2. **Ver estado de la solicitud del mobibus (GET):**



2. Duración real: Realizar una comparación con los tiempos teóricos y los reales del desarrollo para cada etapa del proceso de experimentación

   Se puede ver la comparación implícita en las gráficas arriba.

3. Artefactos construidos: Cuáles artefactos fueron construidos y cuáles no, explicando las causas por las que no fueron desarrollados.

   Los artefactos fueron completamente construidos como se mencionaron.

4. Análisis: Interpreta, explica y justifica los resultados obtenidos basado en patrones y tácticas arquitecturales implementadas en el experimento.

   - Análisis de resultados caso -Mobibus:

     Se sigue cumpliendo con el criterio de calidad ya que las solicitudes son atendidas a una gran velocidad, con un máximo de 480 milisegundos por solicitud. eso es gracias a que el proyecto está en la nube y la cantidad de threads para este tipo de elemento no eran tantas.

   - Análisis de resultados caso Tranvía:

     Se sigue cumpliendo con el criterio de calidad ya que las solicitudes son atendidas a una gran velocidad, con un máximo de 341 milisegundos por solicitud. eso es

gracias a que el proyecto está en la nube y la cantidad de threads para este tipo de elemento no eran tantas.

- Análisis de resultados caso Vcubs:

  Se sigue cumpoliendo con el criterio de calidad ya que las solicitudes son atendidas a una gran velocidad, con un máximo de 262 milisegundos por solicitud. eso es gracias a que el proyecto está en la nube y a pesar de la gran cantidad de elementos que poseía esta clase, las tareas a realizar eran muy sencillas; lo cual hizo que el procesamiento de información no tome mucho tiempo

- Análisis de resultados caso Usuario:

  Se cumple con el criterio de calidad para todos los casos ya que las solicitudes son atendidas a gran velocidad y adicional a esto por lo menos el 99.95% de las solicitudes son atendidas por el sistema.

5. Conclusiones: Teniendo en cuenta la hipótesis, plantea recomendaciones basadas en los resultados obtenidos en la etapa de experimentación.

   Para todos los casos de uso en el sistema planteado, se cumple de manera óptima los criterios de calidad estipulados por nuestro cliente, por lo tanto podemos concluir que gracias a nuestra arquitectura, lógica y recursos el producto final será eficiente cumplira con satisfacer al usuario. Adicional a esto se puede ver que se cumple los atributos de calidad estipulados con las métricas dictadas, es decir que para el experimento 3 primera entrega se ve que se atienden el 99.95% de las solicitudes realizadas. Igualmente el atributo de modificabilidad de se ve completado.

Análisis atributo modificabilidad:

Las siguientes imagenes muestran la actividad del standalone, en estas se puede ver garantizado el atributo de modificabilidad.

SQALE Rating

A

Technical Debt Ratio

1.7%

Debt

3h 42min

Issues

32

Blocker	0
Critical	10
Major	2
Minor	18
Info	2

Directory Tangle Index

0.0%

Cycles

> 0

Dependencies To Cut

Between Directories

0

Between Files

0

Lines Of Code

442

Java

Files

2

Directories

2

Lines

536

Functions

12

Classes

2

Statements

146

Accessors

0

Duplications

0.0%

Lines

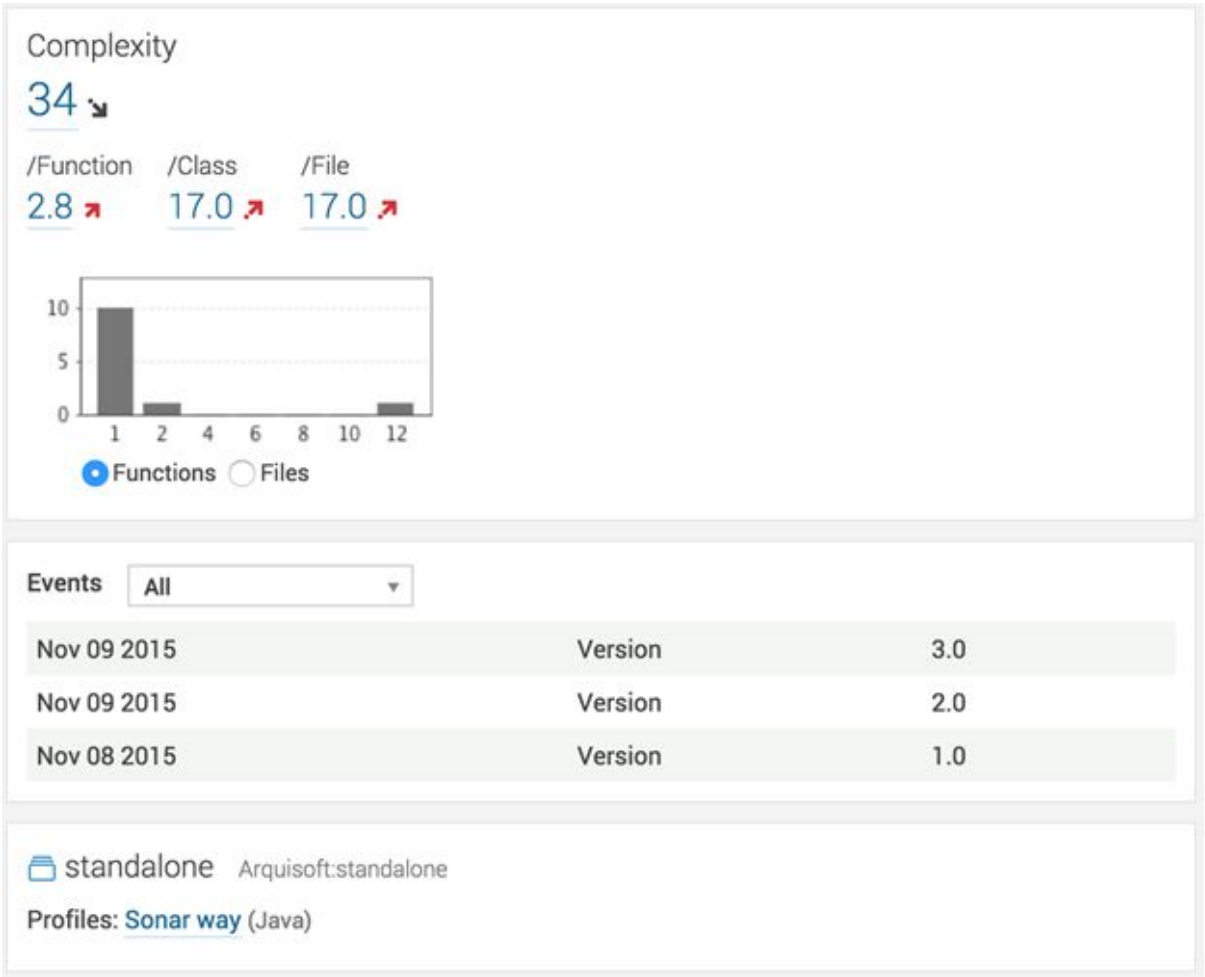
0

Blocks

0

Files

0



Comparación con experimento anterior.

- En este experimento se volvió a utilizar el doble balanceador de carga; y por esto se sigue cumpliendo el atributo de disponibilidad como se desea. Este experimento sigue cumpliendo con los atributos de calidad exigidos en el experimento uno, lo que implica que las modificaciones realizadas no afectan negativamente algún aspecto del experimento. A diferencia del anterior, por medio del artefacto de php sonar se garantizo la modificabilidad del experimento.