Pre-Experimentación:

1. Problemática: El problema principal expuesto en el enunciado es la dificultad del control oportuno de los vehículos en su flota. Para esto se propone que utilizando el dispositivo GPS integrado en cada uno de los vehículos (excepto los vcubs), se reporte la localización actual de los vehículos al sistema central, quien se encargará de tomar decisiones en base a las necesidades actuales. Además de esto se plantea que las estaciones de vcub den su porcentaje de ocupación. Todo esto se plantea hacer en real time.
2. Objetivo: Se plantea utilizar PhP lumen para implementar el sistema de recepción de información en real time tanto de las estaciones como de los vehículos. Y en el aviso de emergencia de los tranvías. Una vez recibida la información se utiliza la arquitectura dicha para tomar decisiones de las rutas de los mobibuses, llenado nuevo de las estaciones de vcubs y supervisión del sistema de tranvías.
3. Descripción del experimento: Las actividades a realizar son:

- Desarrollar la capa de lógica de negocio que debe ofrecer el servidor del centro de control.
- Desarrollar las pruebas de carga con loader.io.
- Realizar la versión final del documento de experimentación.
- Desarrollar la capa de persistencia.
- Desarrollar y poblar la base de datos.
- Analizar y comparar los resultados obtenidos en la entrega parcial con los de la entrega final del experimento según los documentos.
- Desarrollar la capa de presentación de la aplicación Web requerida por el operador de Tbc.
- Desarrollar la capa de presentación de las aplicaciones Standalone que simulan Mobibus, VCub, Tranvía.

Y los datos a recolectar son:

- Datos de latencia: tiempo de respuesta de la aplicación.
- Datos de escalabilidad: tiempo de soporte de x cantidad de vehículos.

4. Los artefactos a construir inicialmente son: los controladores de las entidades básicas (Mobibus, tranvía, usuario y estación de vcub), el ruteo de dichos elementos y lo mínimo para que los requerimientos CRUD se cumplan adecuadamente. Para la versión final se deben construir: la base de datos y su respectiva población, los standalones para las simulaciones, la interfaz grafica del usuario y el medio para la conexión de estos.
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.

6. Los resultados esperados son:

- Que el tiempo de respuesta de la aplicación con 4500 elementos sea de un segundo o menos.

- Que el tiempo de soporte sea de 5 segundos máximo con 4500 vehículos en el sistema.

7. El tiempo esperado para la implementación de la capa lógica es de aproximadamente 4 horas de trabajo en lenguaje PhP por integrante, encima de esto 2 horas de todo el grupo para identificación de los casos de usos y requerimientos del experimento y 1 hora para la realización de este documento.
Para la realización de la entrega final se piensan los siguientes tiempos: 1.5 horas para la construcción del front end la aplicación web, 1 hora para la realización de la base de datos y su población, 1 hora para la construcción de la aplicación stand alone, 1 hora para la capa de persistencia y 0.5 horas para la realización de pruebas y la fabricación de este documento.
Post-Experimentación

RESULTADOS OBTENIDOS:

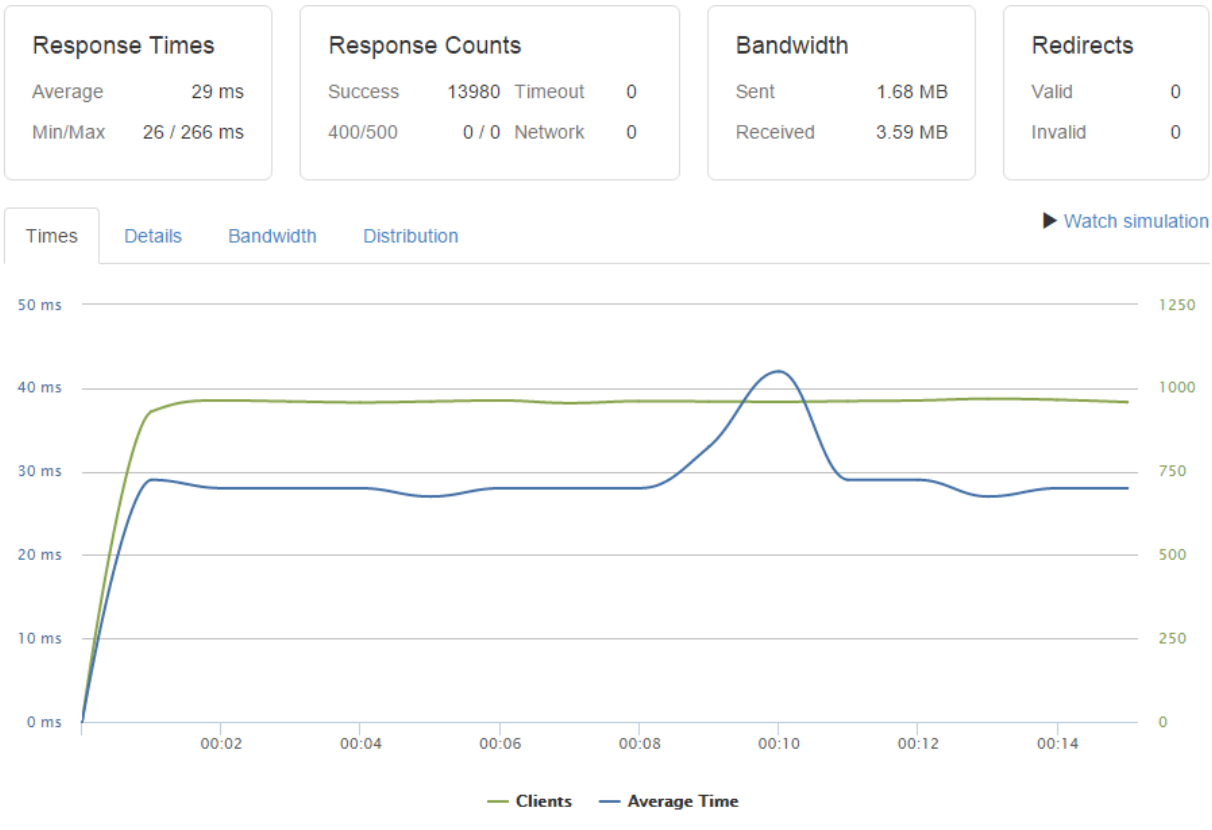
Los resultados obtenidos fue la construcción de la lógica de la aplicacion en lenguaje php en un tiempo estimado de 5 horas por integrante. Identificando 12 casos de uso.

Análisis de Resultados:

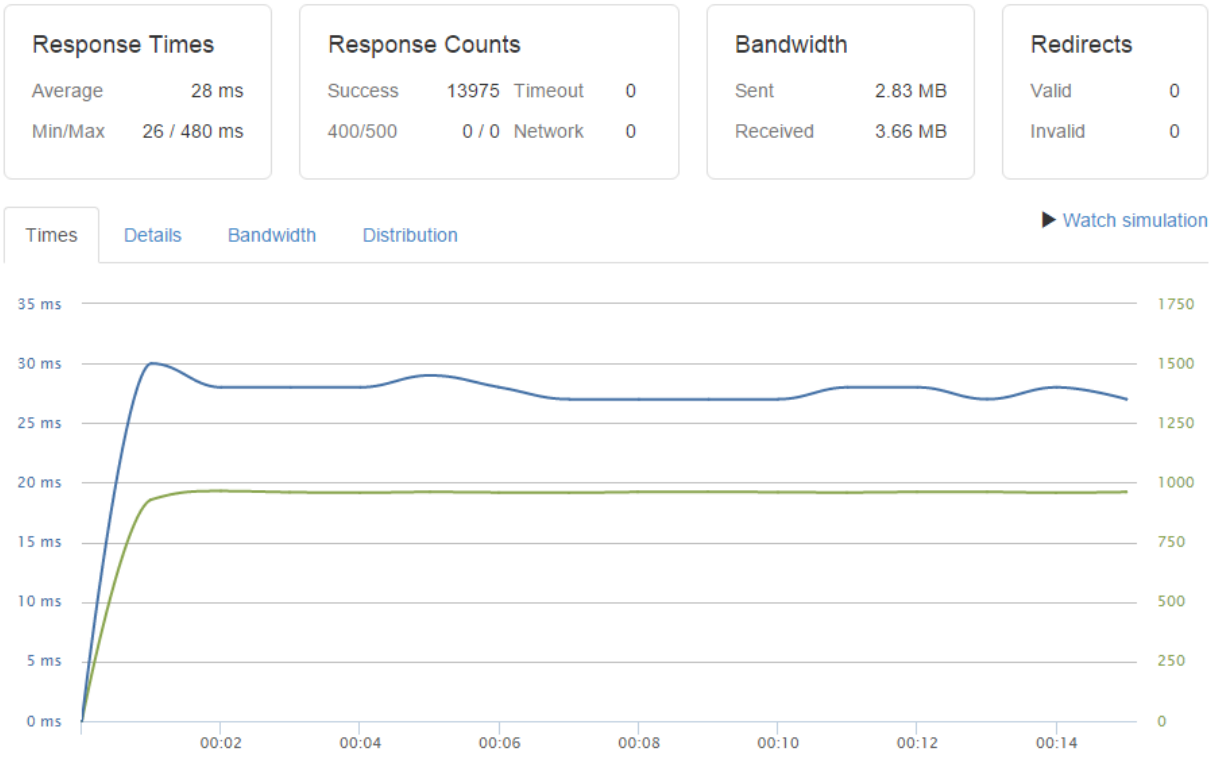
Pruebas de latencia - Casos de uso:

CASOS PARA EL MOBIBUS:

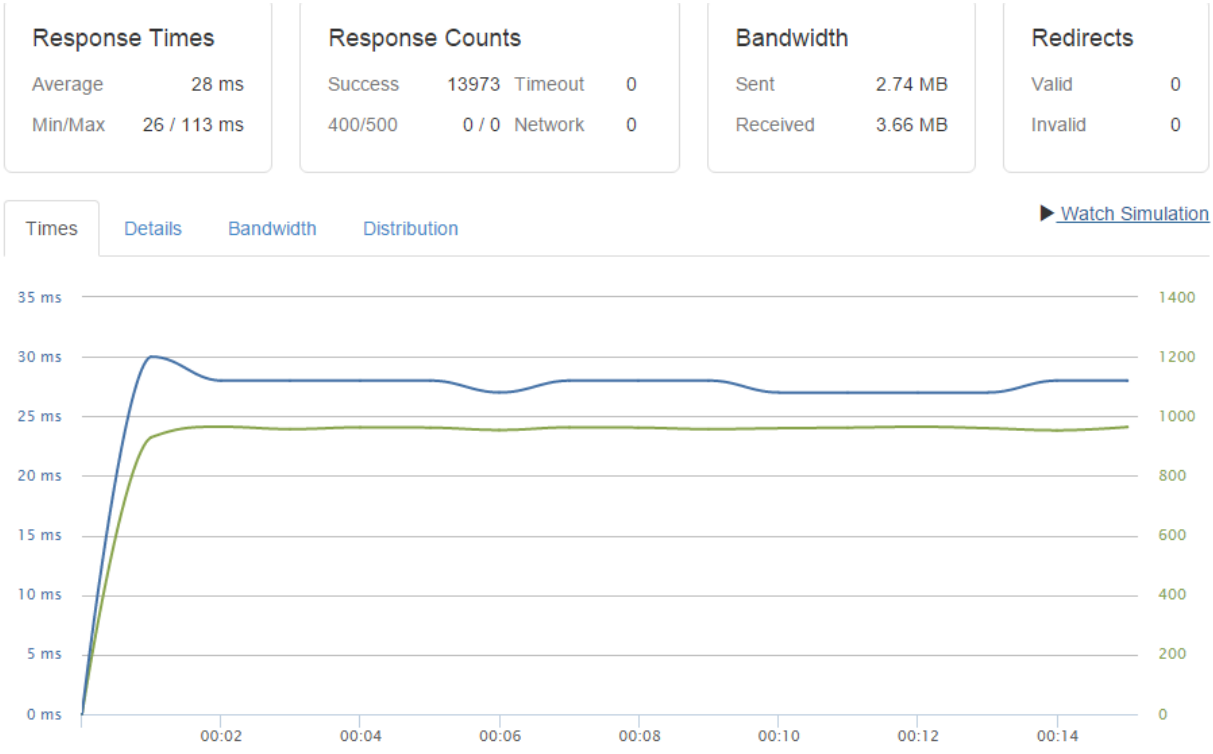
1. Ver un mobibus (GET):



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



3. Ocupar un Mobibus (POST)

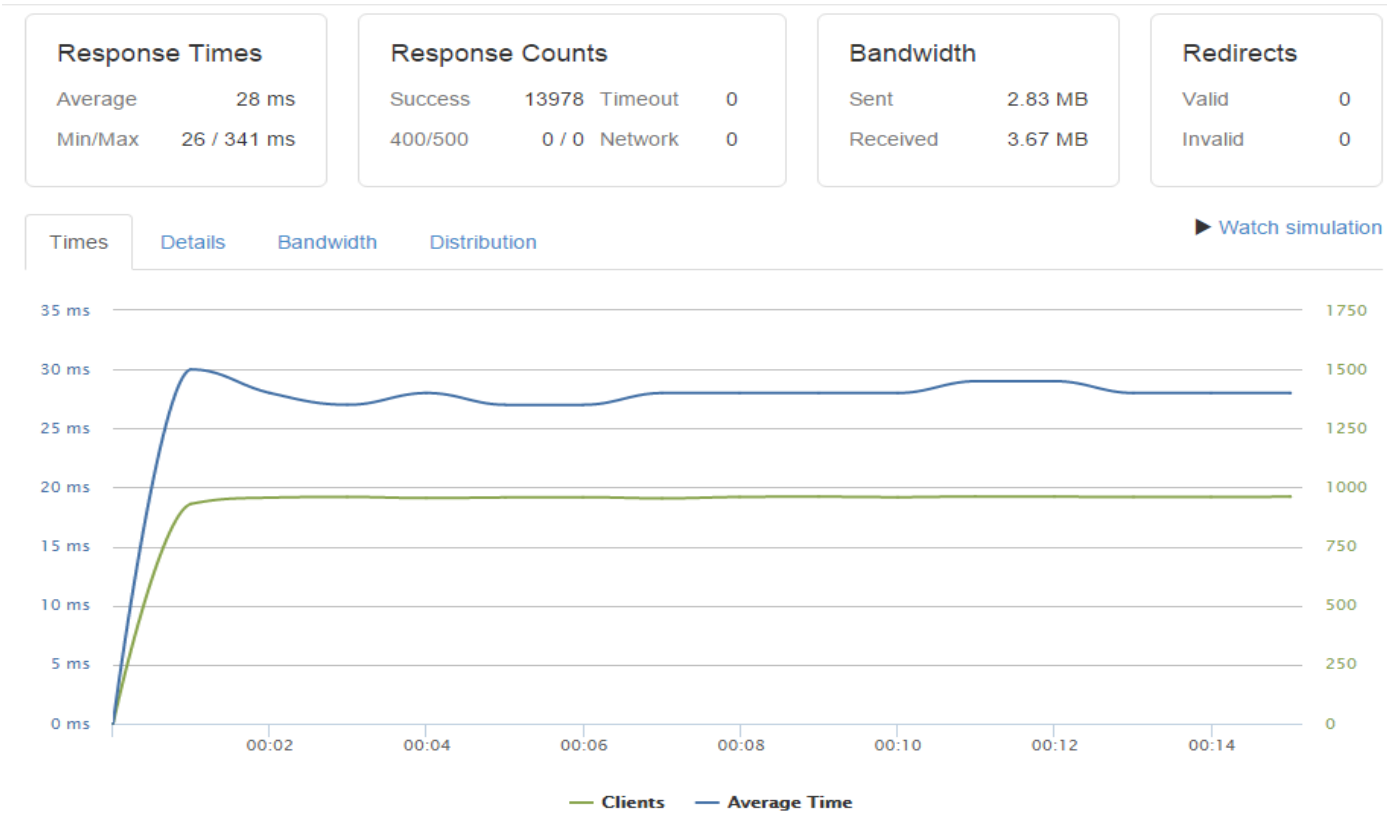


Analisis de resultados caso -Mobibus:

Se cumple 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.

CASOS PARA TRANVÍA:

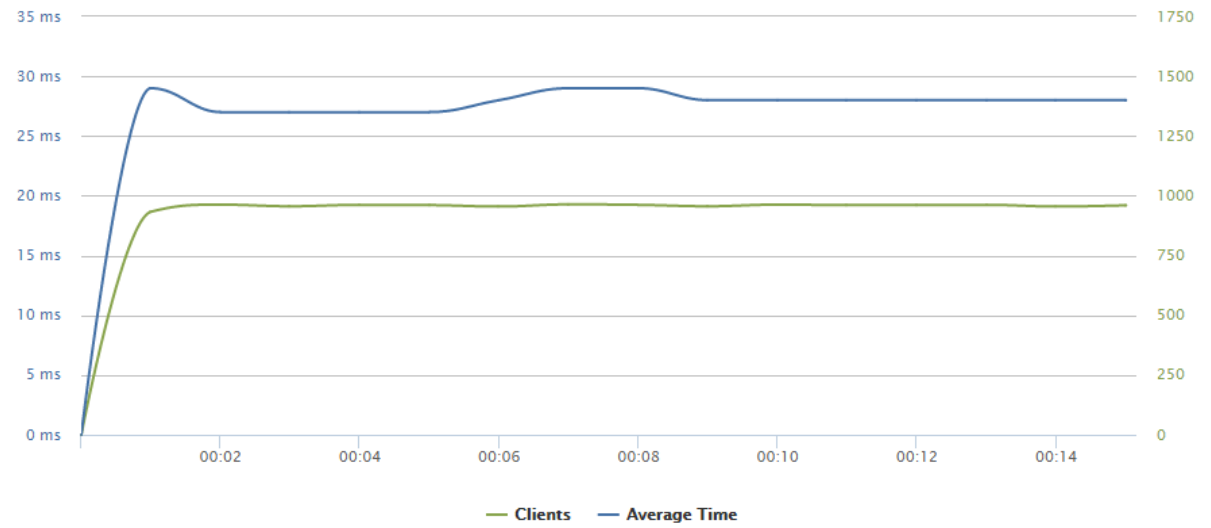
1. Obtener un tranvía (GET):



2. Reportar La posicion (POST)

Response Times		Response Counts				Bandwidth		Redirects	
Average	28 ms	Success	13975	Timeout	0	Sent	2.75 MB	Valid	0
Min/Max	26 / 251 ms	400/500	0 / 0	Network	0	Received	3.62 MB	Invalid	0

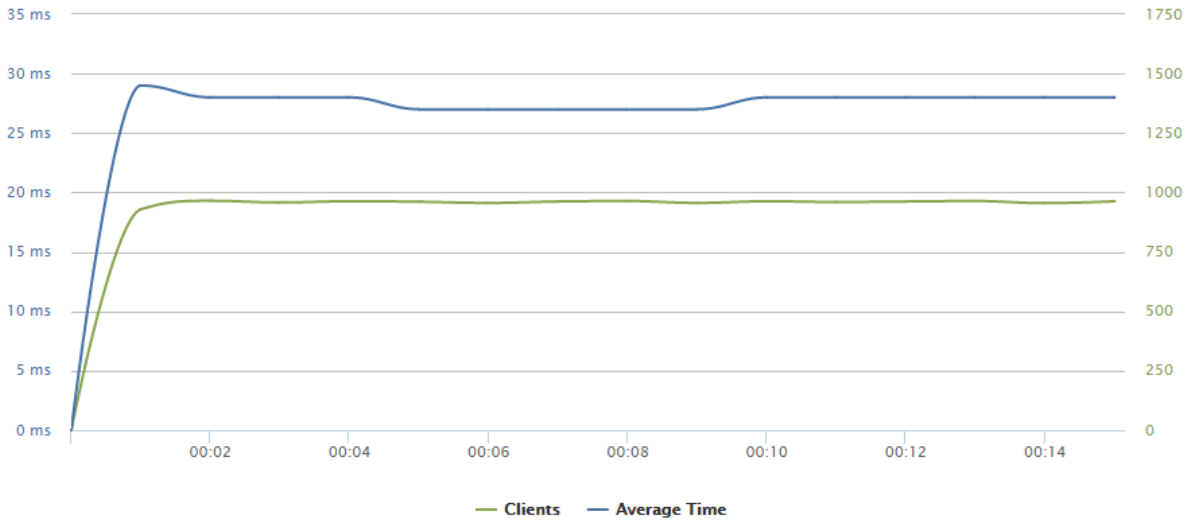
Times Details Bandwidth Distribution ▶ Watch simulation



3. Reportar Emergencia

Response Times		Response Counts				Bandwidth		Redirects	
Average	28 ms	Success	13976	Timeout	0	Sent	1.68 MB	Valid	0
Min/Max	26 / 255 ms	400/500	0 / 0	Network	0	Received	3.48 MB	Invalid	0

Times Details Bandwidth Distribution ▶ Watch simulation

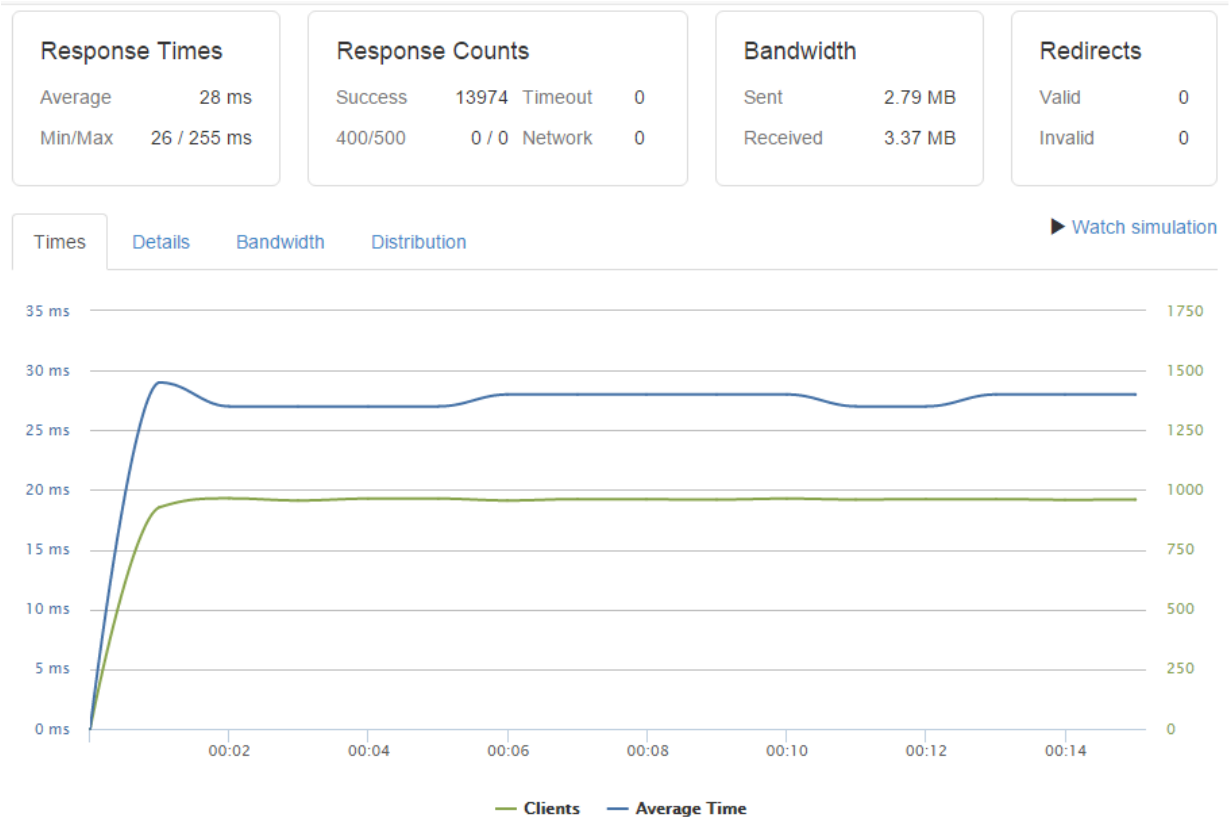


Analisis de resultados caso Tranvía:

Se cumple 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.

CASOS PARA VCUBS:

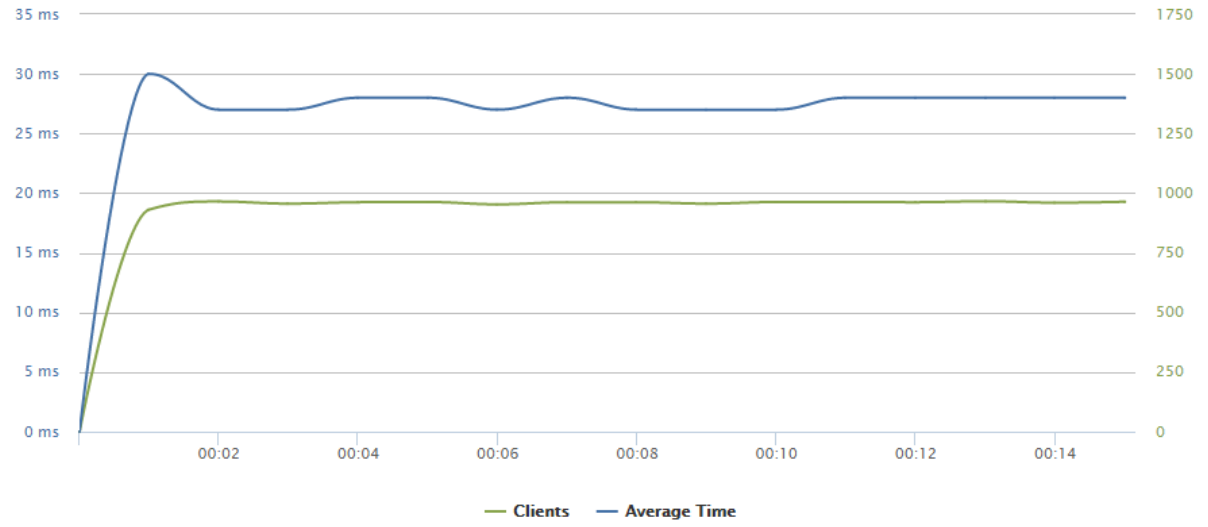
1. Registrar un Vcub en una terminal (POST):



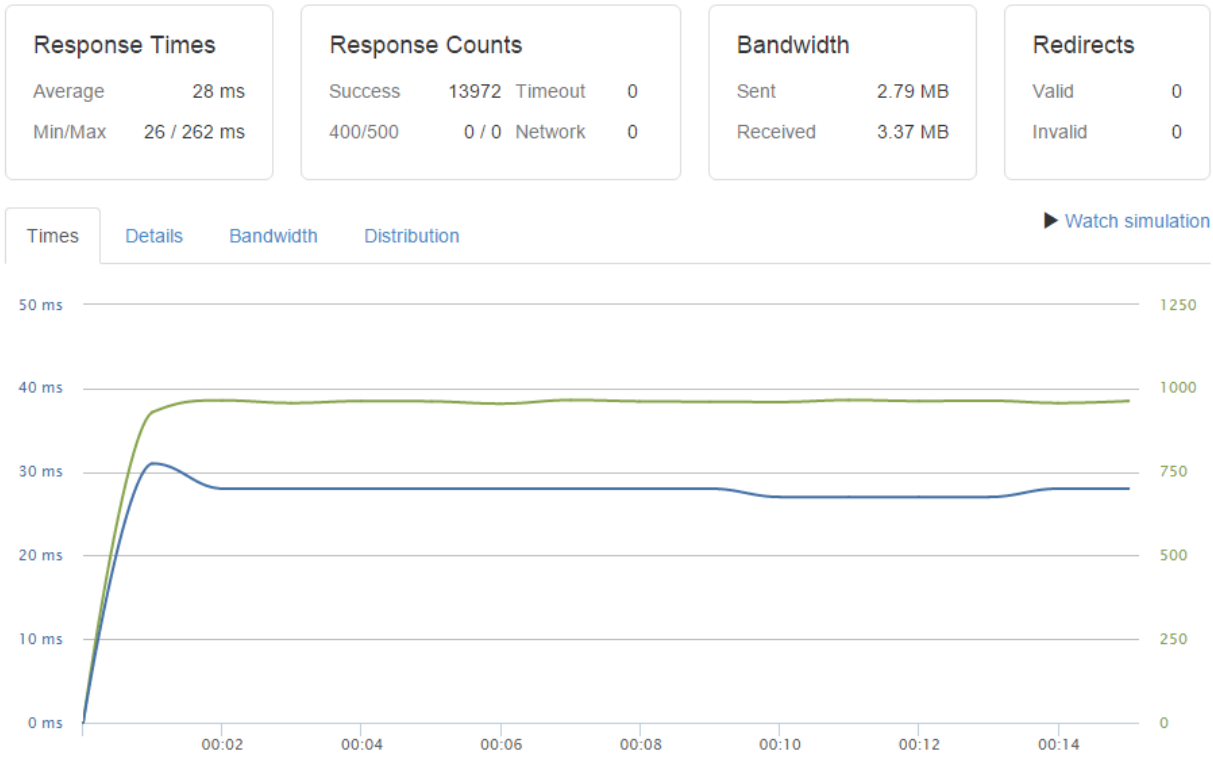
2. Prestar un Vcub (POST):

Response Times		Response Counts			Bandwidth		Redirects	
Average	28 ms	Success	13973	Timeout 0	Sent	2.80 MB	Valid	0
Min/Max	26 / 260 ms	400/500	0 / 0	Network 0	Received	3.34 MB	Invalid	0

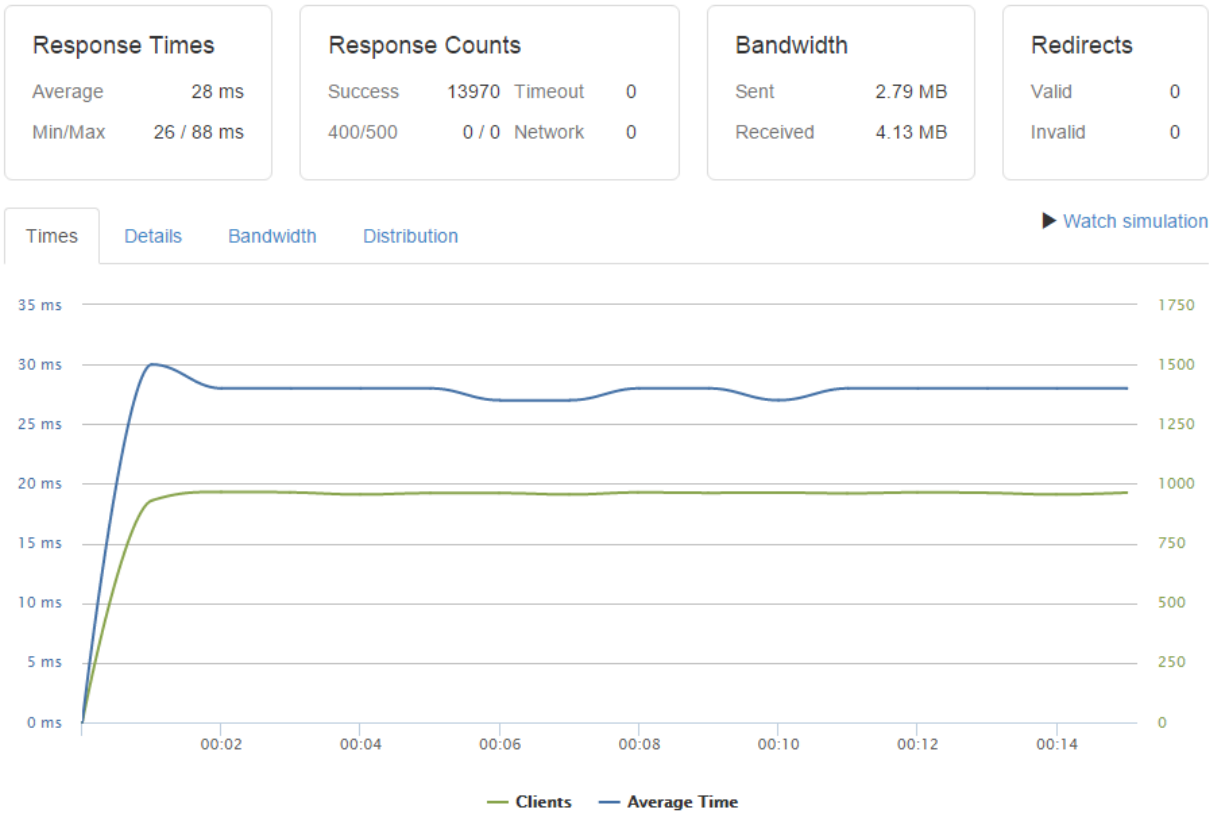
Times Details Bandwidth Distribution ▶ Watch simulation



3. Recibir un Vcub (PUT):



4. Pedir un llenado de Vcubs:

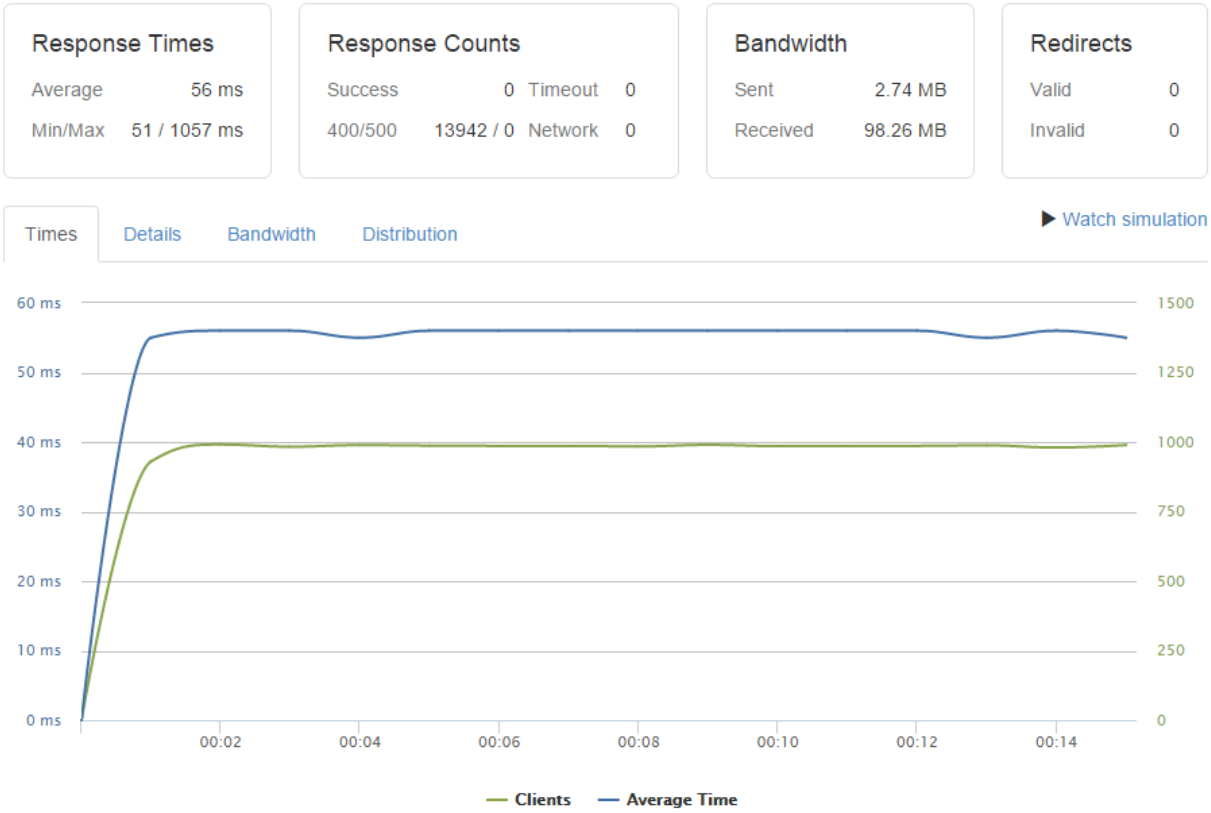


Análisis de resultados caso Vcubs:

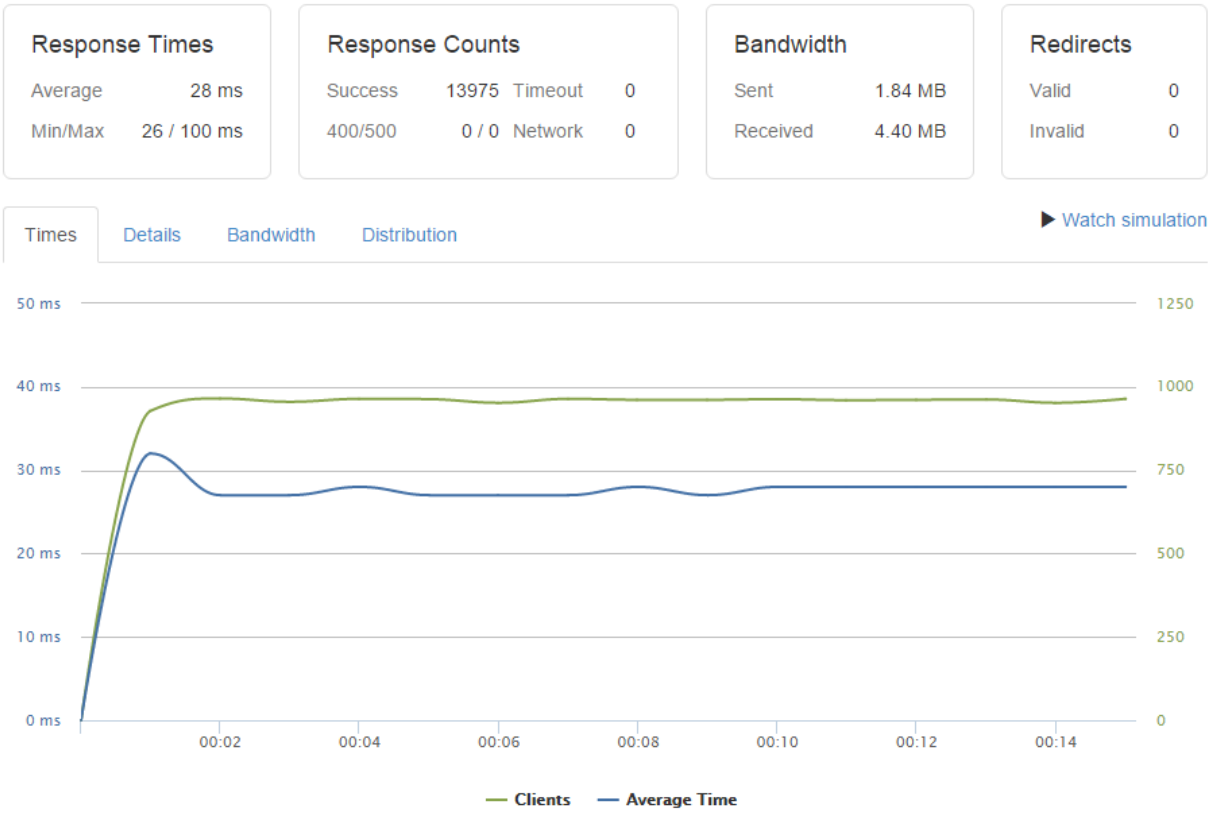
Se cumple con el criterio de calidad ya que las solicitudes son atendidas a una gran velocidad, con un máximo de 262 milisegundos por solicitud. eso es gracias a que el proyecto está en la nube y a pesar de la gran cantidad de elementos que poseía esta clase, las tareas a realizar eran muy sencillas;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

CASOS PARA Usuario:

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



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



Análisis de resultados caso Usuario:

No cumple con el criterio de calidad para todos los casos ya que a pesar de que las solicitudes son atendidas a gran velocidad, el caso en el que se solicita un mobibus tiene un máximo de 1057 milisegundos, 57 milisegundos más lento de lo que se esperaba, esto se debe a que hay un análisis de todos los mobibuses cerca del área, esta información se compara con la latitud y la longitud en la que se encuentra el usuario y por lo tanto hay que revisar la posicion de todos los mobibuses en el sistema.

Conclusion final:

Para casi 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.

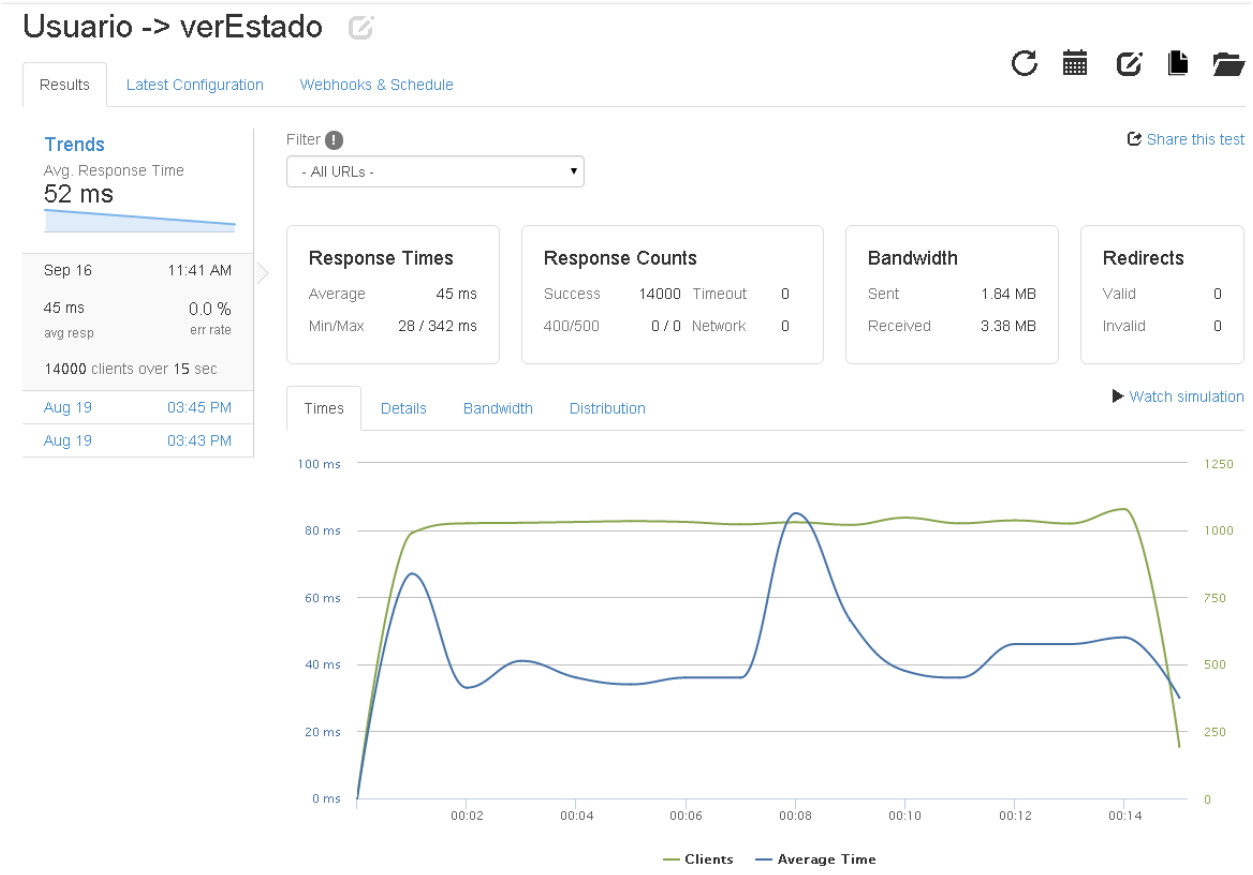
Experimento 1: Segunda parte

En esta ocasión se integra el sistema con una aplicación tanto web como standalone integrada a la lógica presentada en la primera entrega. Esta se presentará en la web ya que estas aplicaciones se localizan en la nube.

Para esto probaremos los casos de uso en las aplicaciones por lo que mostraremos gráficas de los casos de uso en la aplicación, comparada solo con la lógica.

Caso de uso:

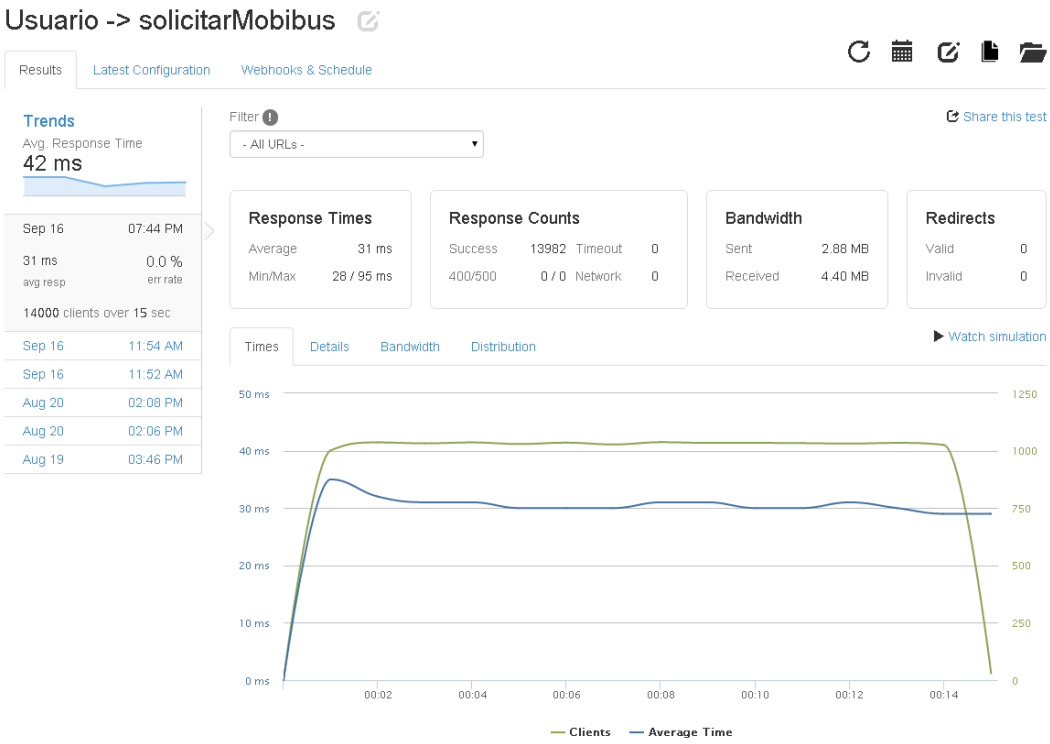
Usuario ver estado de una solicitud de mobibus:



VERSIÓN APLICACIÓN.

Si comparamos con la versión de lógica podemos ver claramente una fluctuación en el tiempo promedio de respuesta con respecto a una constante de cliente. Esto muestra un ámbito más realista de lo que puede pasar en una aplicación web dada al público. A pesar de esto se cumple el requerimiento de realizar satisfactoriamente 1400 requerimientos en menos de 1 segundo.

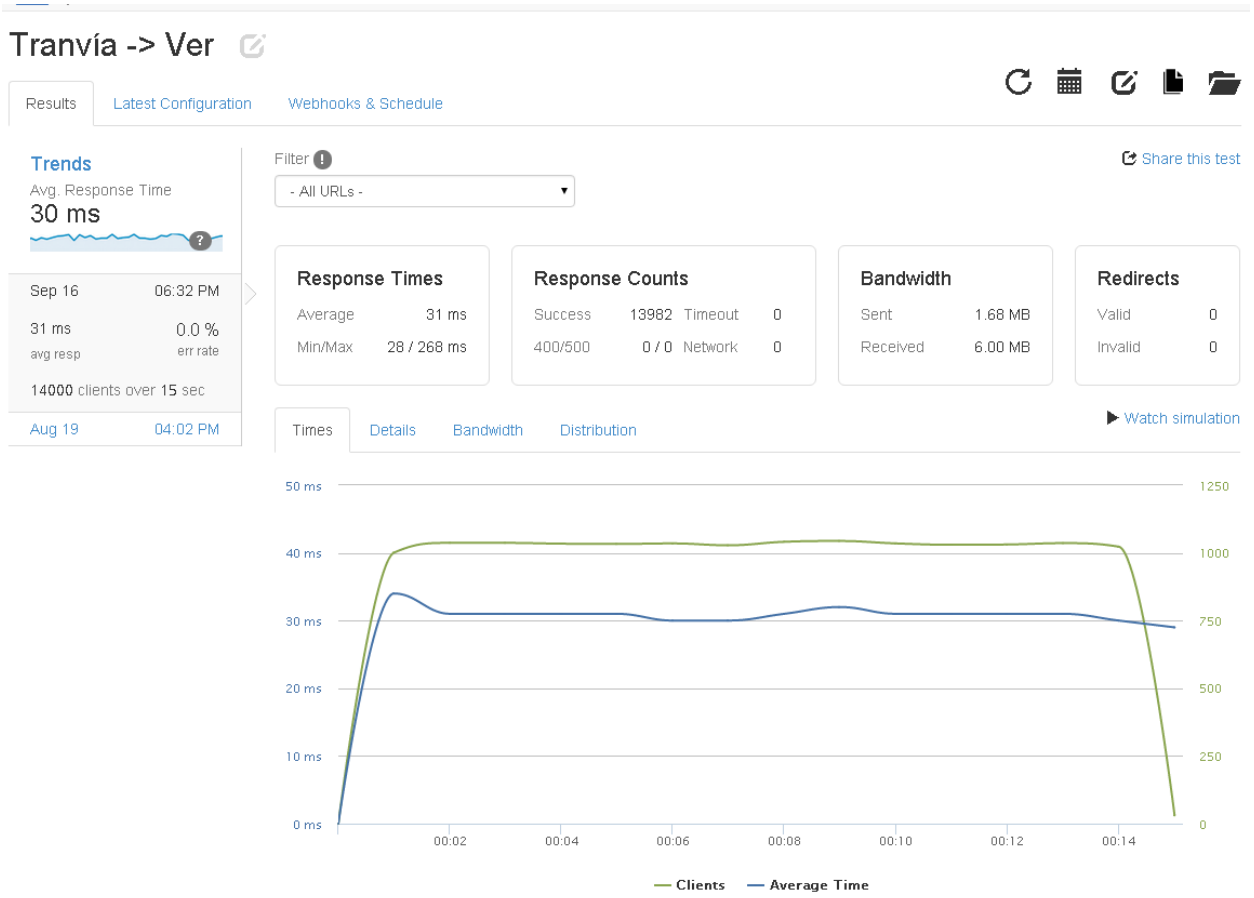
Caso de uso:
Usuario: Solicitar Mobibus.



Como podemos apreciar, en este caso es un tiempo de respuesta más estable comparado con el número de clientes que es constante. Sin embargo, comparado con el modelo de lógica se muestra que el tiempo de respuesta es menor. Esto se debe a que el acceso a la base de datos por solicitud es continua. Esto también explica el fallo al procesar satisfactoriamente 13982 solicitudes, puesto que dada la gran cantidad de solicitudes el sistema falla.

Caso de uso

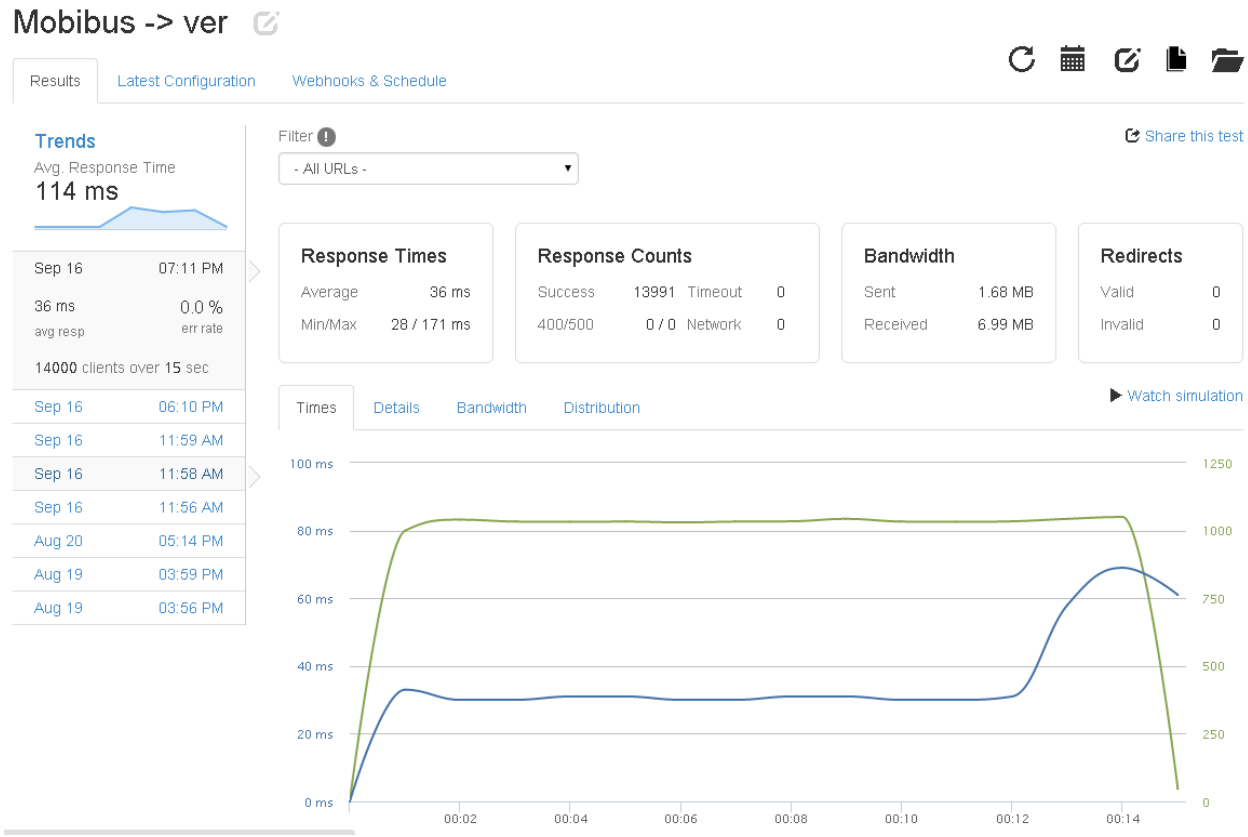
Tranvia: Ver información de un tranvía.



En este caso, igual que el anterior el tiempo de respuesta es menor comparado solo con el módulo de lógica. También, dada la gran cantidad de solicitudes a la base de datos muestra que falla en responder satisfactoriamente 18 solicitudes.

Caso de uso:

Mobibus: ver información



En este caso se ve un gran desfalco en el tiempo de respuesta comparado con el modulo de logica. Esto se debe a que cada vez que se ingresa una solicitud, el sistema revisa toda la tabla de los mobibus en búsqueda de el elemento que cumpla con los parametros a buscar. Este, igual que los casos anteriores, falla en cumplir las 1400 solicitudes en menos de un segundo por la misma razón presentada en los puntos anteriores de este experimento.

Conclusión:

La implementación de esta aplicación mostrará ciertos desfalcos en términos de tiempo de respuesta por cantidad de solicitudes. Muestra errores típicos en términos de responder las solicitudes (como fue anteriormente mostrado) para una cantidad mínima de solicitudes. Esta aplicación sin embargo cumple en su mayor parte tanto con los requerimientos de calidad como con los casos de uso solicitados ya que estos tienen como base la capa de lógica presentada anteriormente que cumple en su totalidad con los requerimientos solicitados. Puede presentar mejoras a lo largo del tiempo ya que es fácil de modificar dependiendo de los nuevos casos a presentar.