Documentación de Arquitectura
Caso de estudio TBC

Grupo 04
Yammz
Carlos Felipe Agudelo
Samuel Baquero
Gabriel Martinez
Luis Miguel Mejía
Sergio Yodeb Velásquez

Arquitectura y diseño de Software
Kelly Johany Garcés Pernett
Gilberto Pedraza

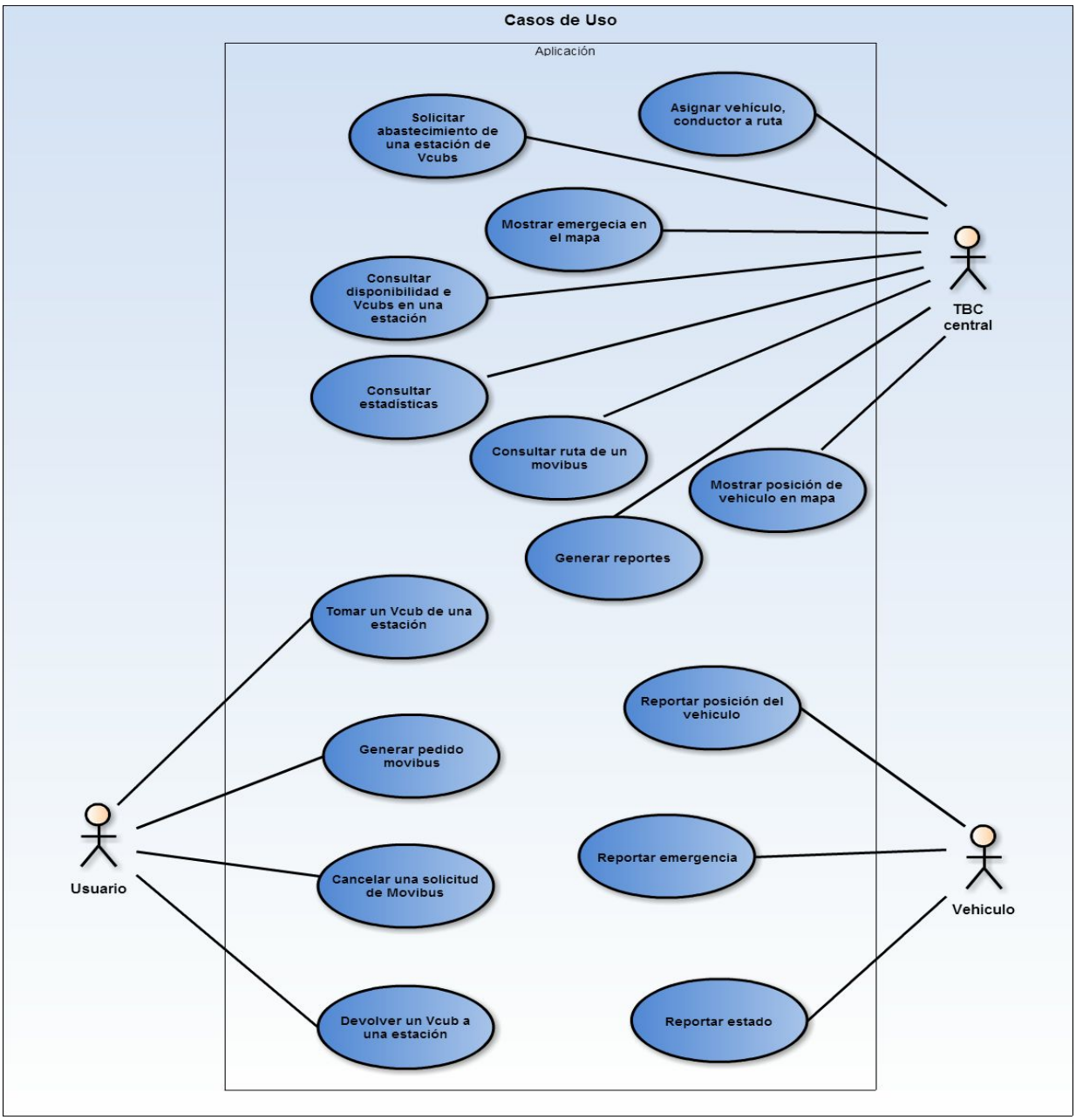
Bogotá D.C.
2015-2

Contenido

1. Restricciones

Identificador	Tipo	Descripción
Re1	Tecnologías	Las aplicaciones embebidas en los vehículos (Mobibus, tranvía) y estaciones de VCub tienen que ser simuladas con aplicaciones standalone.

2. Casos de uso



3. Atributos de calidad

Desempeño: La restricción de desempeño que la arquitectura planteada satisfará que todas las solicitudes y consultas de la aplicación (ver Casos de Uso) serán satisfechas en 1 segundo o menos. Para cumplir con este requerimiento no funcional se plantea una arquitectura con actores, que permite gracias a la interacción del actor consumidor, productor e intermediario aumentar el desempeño de toda la aplicación. Se optimizarán también los algoritmos que rigen a la lógica y sus servicios.

Escalabilidad: La restricción de escalabilidad que la arquitectura es que el sistema soportará el envío y la lectura de la información de 4,500 vehículos en un espacio de 5 segundos. En busca de satisfacer este requerimiento no funcional, se hará uso de la escalabilidad que ofrece la arquitectura con actores en el framework Play; esto debido a que este framework cuenta con la capacidad de aumentar los actores que participan en ella dependiendo de la demanda sobre la cual se encuentre el sistema. Es además posible configurar un mayor uso de los recursos de la máquina sobre la que se ejecuta la aplicación al aumentar el pool de los threads disponibles para ejecutar las respuestas a las solicitudes, y tener así respuesta a las peticiones de forma más rápida y efectiva.

4. Escenarios de calidad

- **Desempeño**

Identificador	**Tipo**	**Prioridad**
EC1	Desempeño	Alta
Fuente		
TBC (Vehículos, Estaciones de Vcubs), Usuario.		
Estímulo		
Solicitud a uno de los servicios de la aplicación.		
Ambiente		
Explotación, estrés.		
Medida esperada		
Una solicitud cualquiera deberá ser respondida en 1 segundo o menos		

Identificador	**Tipo**	**Prioridad**
EC2	Desempeño	Alta
Fuente		
Vehículo		
Estímulo		
Alarma o cambio repentino de estado en un vehículo.		
Ambiente		
Normal		
Medida esperada		

El sistema debe recibir la alarma o cambio de estado y reportarla a la central en menos de 1 segundo.

Identificador	**Tipo**	**Prioridad**
EC3	Desempeño	Baja
Fuente		
Usuario		
Estímulo		
Cancelar pedido de Movibus		
Ambiente		
Normal		
Medida esperada		
El sistema debe cancelar un pedido de movibus en menos de 1 segundo.		

Identificador	**Tipo**	**Prioridad**
EC4	Desempeño	Alta
Fuente		
Usuario		
Estímulo		
Realizar pedido de Movibus.		
Ambiente		
Saturado		
Medida esperada		
El sistema debe responder a un pedido de Movibus en menos de 1 segundo.		

Identificador	**Tipo**	**Prioridad**
EC5	Desempeño	Alta
Fuente		
Vehículos		

Estímulo
Envío de reporte de emergencia.
Ambiente
Normal.
Medida esperada
El reporte debe ser procesado en 1 segundo o menos.

Identificador	**Tipo**	**Prioridad**
EC6	Desempeño	Alta
Fuente		
TBC		
Estímulo		
Solicitar que se muestre en un mapa la posición de los vehículos y las emergencias.		
Ambiente		
Saturado.		
Medida esperada		
El mapa debe ser generado en menos de 1 segundo.		

Identificador	**Tipo**	**Prioridad**
EC7	Desempeño	Media
Fuente		
TBC		
Estímulo		
Generar estadísticas de datos del sistema y generar reportes de estos.		
Ambiente		
Normal.		
Medida esperada		
El reporte debe ser generado en menos de 3 segundos.		

Identificador	Tipo	Prioridad
EC8	Desempeño	Media
Fuente		
TBC(Estaciones de Vcubs)		
Estímulo		
Solicitar el abastecimiento de Vcubs a una estación.		
Ambiente		
Normal.		
Medida esperada		
La petición debe ser procesada en 1 segundo o menos.		

- **Escalabilidad**

Identificador	Tipo	Prioridad
EC9	Escalabilidad	Alta
Fuente		
TBC (Vehículos, Estaciones de Vcubs).		
Estímulo		
Solicitud a uno de los servicios de la aplicación.		
Ambiente		
Explotación, estrés.		
Medida esperada		
4,500 solicitudes a la aplicación deberá ser respondidas a totalidad y sin errores en 5 segundos o menos.		

Identificador	Tipo	Prioridad
EC10	Escalabilidad	Media
Fuente		
Usuario		

Estímulo
Tomar Vcub de una estación.
Ambiente
Saturado
Medida esperada
El sistema debe responder 500 solicitudes de préstamo de Vcubs en menos de 5 segundos.

Identificador	**Tipo**	**Prioridad**
EC11	Escalabilidad	Media
Fuente		
Usuario		
Estímulo		
Devolver Vcub a una estación.		
Ambiente		
Saturado		
Medida esperada		
El sistema debe responder 500 solicitudes de devolución de Vcubs en menos de 5 segundos.		

Identificador	**Tipo**	**Prioridad**
EC12	Escalabilidad	Alta
Fuente		
Vehículo		
Estímulo		
Reporte de posición de un vehículo.		
Ambiente		
Saturado		
Medida esperada		

El sistema debe recibir 500 reportes de posición de los Movibuses y tranvías en menos de 5 segundos.

Identificador	**Tipo**	**Prioridad**
EC13	Escalabilidad	Alta
Fuente		
Vehículo		
Estímulo		
Reporte de estado de un vehículo.		
Ambiente		
Saturado		
Medida esperada		
El sistema debe recibir 500 reportes de posición de los Movibuses y tranvías en menos de 5 segundos.		

Identificador	**Tipo**	**Prioridad**
EC14	Escalabilidad	Media
Fuente		
TBC (Vehículos, Estaciones de Vcubs).		
Estímulo		
Asignación de mobibus y conductor a un pedido a una ruta.		
Ambiente		
Normal.		
Medida esperada		
El sistema debe recibir 500 asignaciones de mobibus, conductor a una ruta en menos de 5 segundos.		

Identificador	**Tipo**	**Prioridad**
EC15	Escalabilidad	Media
Fuente		

Usuario
Estímulo
Solicitar la información de la ruta de un movibus.
Ambiente
Saturado
Medida esperada
El sistema debe responder 500 solicitudes de ruta de movibus en menos de 5 segundos.

Identificador	**Tipo**	**Prioridad**
EC16	Escalabilidad	Media
Fuente		
Usuario, TBC		
Estímulo		
Consultar la cantidad de Vcubs en una estación.		
Ambiente		
Saturado		
Medida esperada		
El sistema debe responder 500 consultas de estado de estación de Vcubs en menos de 5 segundos.		

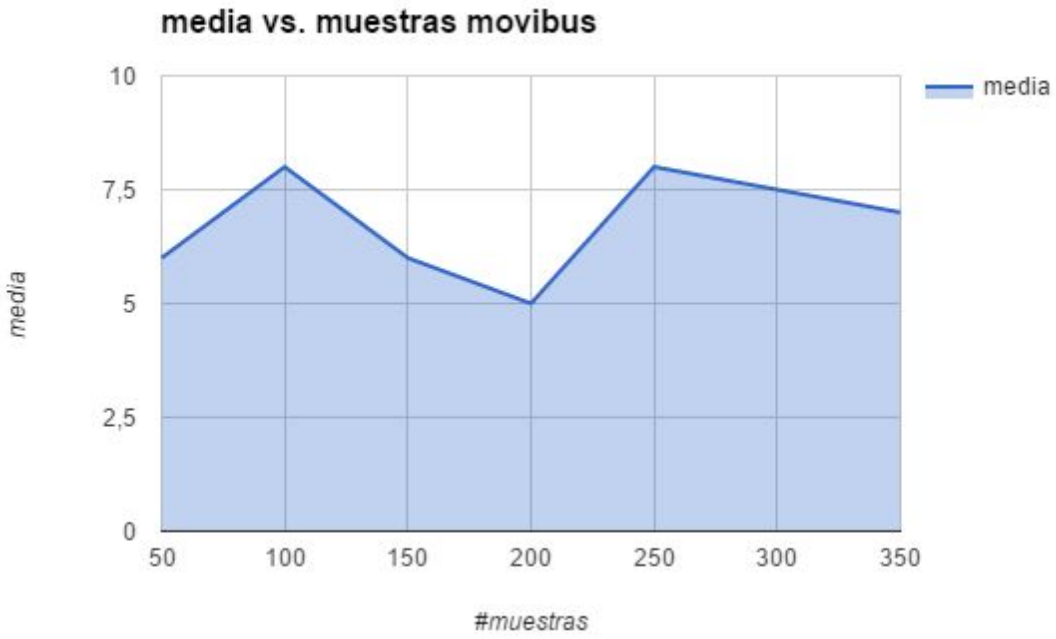
5. Resumen de métricas

Escenario de Calidad	**Atributo de Calidad**	**Métrica**	**Valor Esperado**	**Valor más alto obtenido**
Escenario 1	Desempeño	Latencia	1 segundo	0,512 segundos
Escenario 2	Escalabilidad	Tiempo de respuesta para 4500 informes que se envían en 5 segundos	1 segundo	0.687 segundos

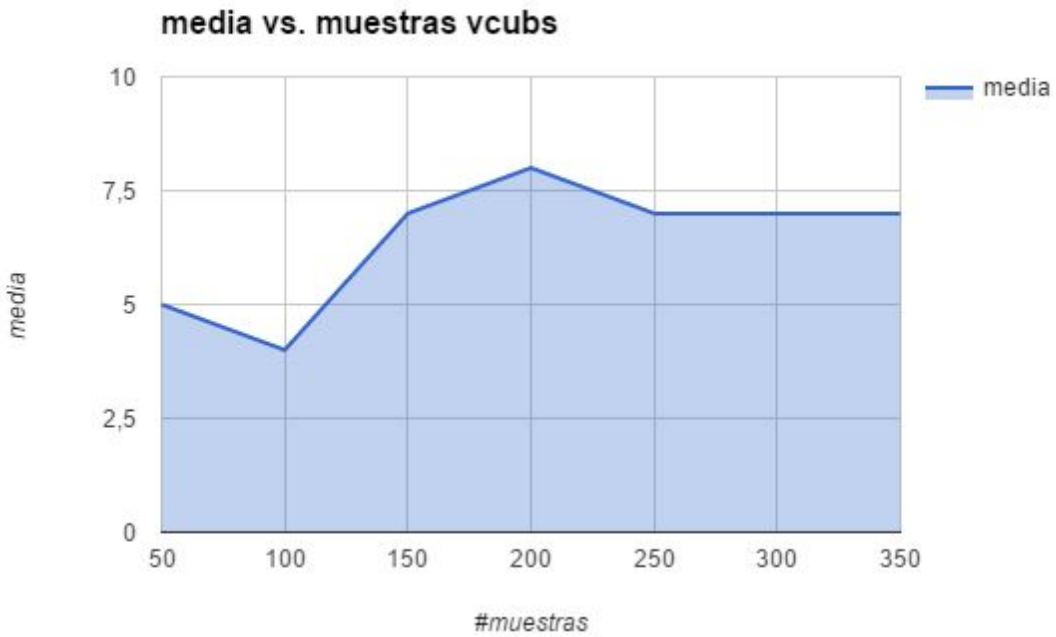
A continuación mostramos algunos ejemplo de los servicios que se realizaron para explicar de forma más precisa los resultados de las métricas. Primero está un servicio simple como

lo es un Get y luego un servicio que implica otras relaciones como lo es reportar accidente tranvia.

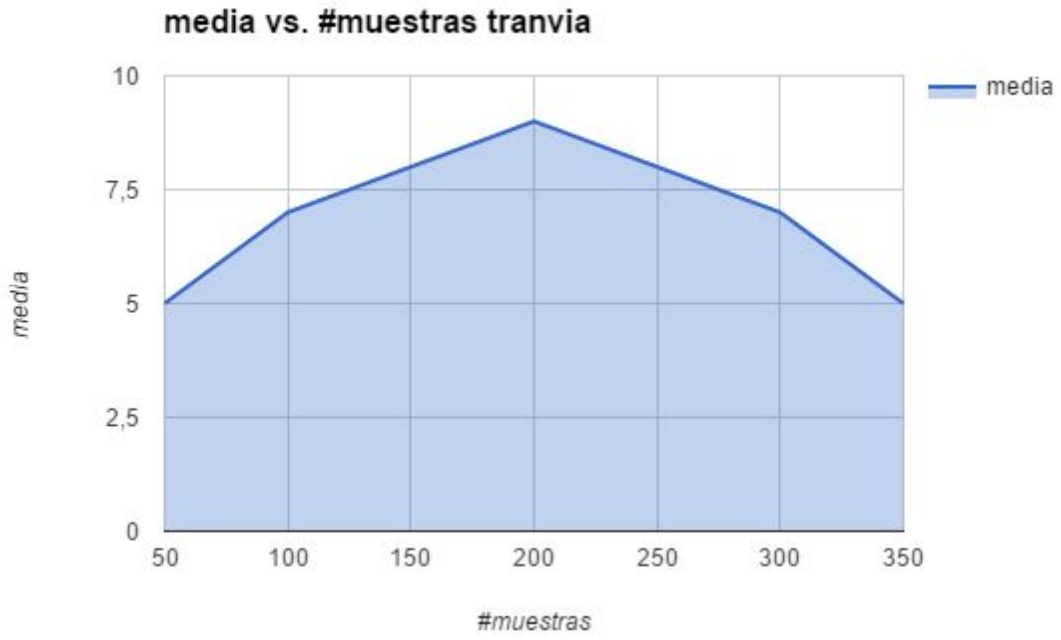
1. **Gráfica 1 servicio get movibus.**



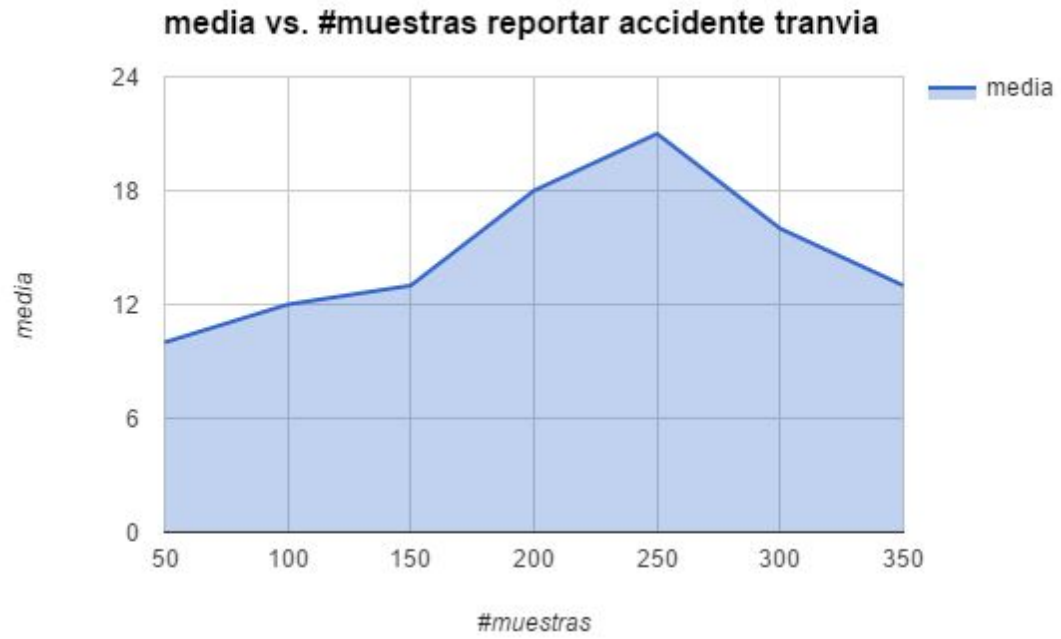
2. **Gráfica 1 servicio get vcu**



3. **Gráfica 3 servicio post tranvía.**



Como se puede ver en las diferentes gráficas de los servicio de get, la escalabilidad de la aplicación va a ser posible gracias a la arquitectura implementada. En la gráfica número uno y en la gráfica número tres se aprecia como en el momento que comienza a aumentar la media del tiempo de respuesta al aumentar el número de solicitudes se crean nuevos actores para disminuir este tiempo y poder cumplir con la demanda.



En esta vemos que aunque el tiempo de respuesta es mayor tiende a tener el mismo comportamiento ya descrito.

6. Vistas arquitecturales

Diagrama de Contexto

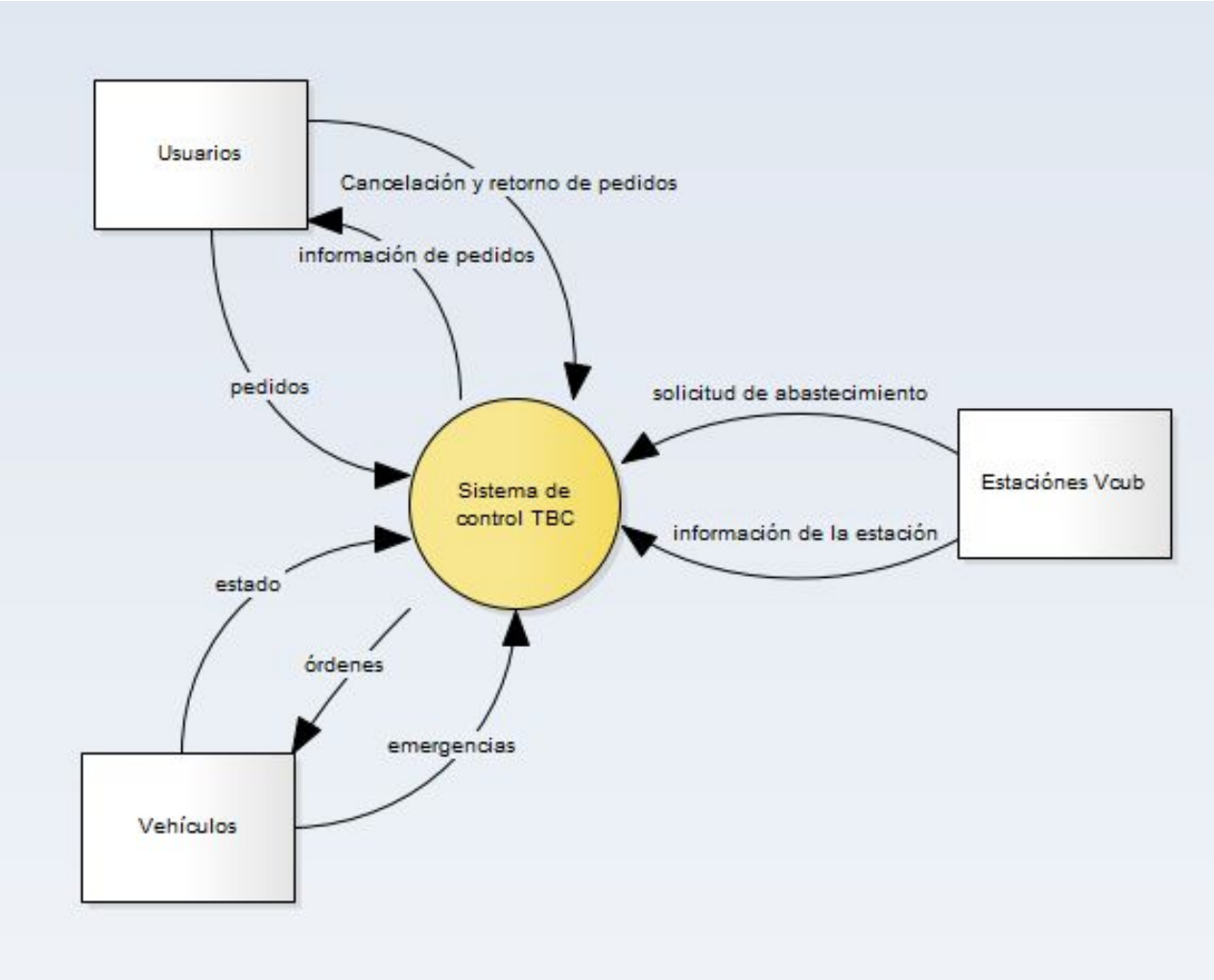


Diagrama de entidad/relación

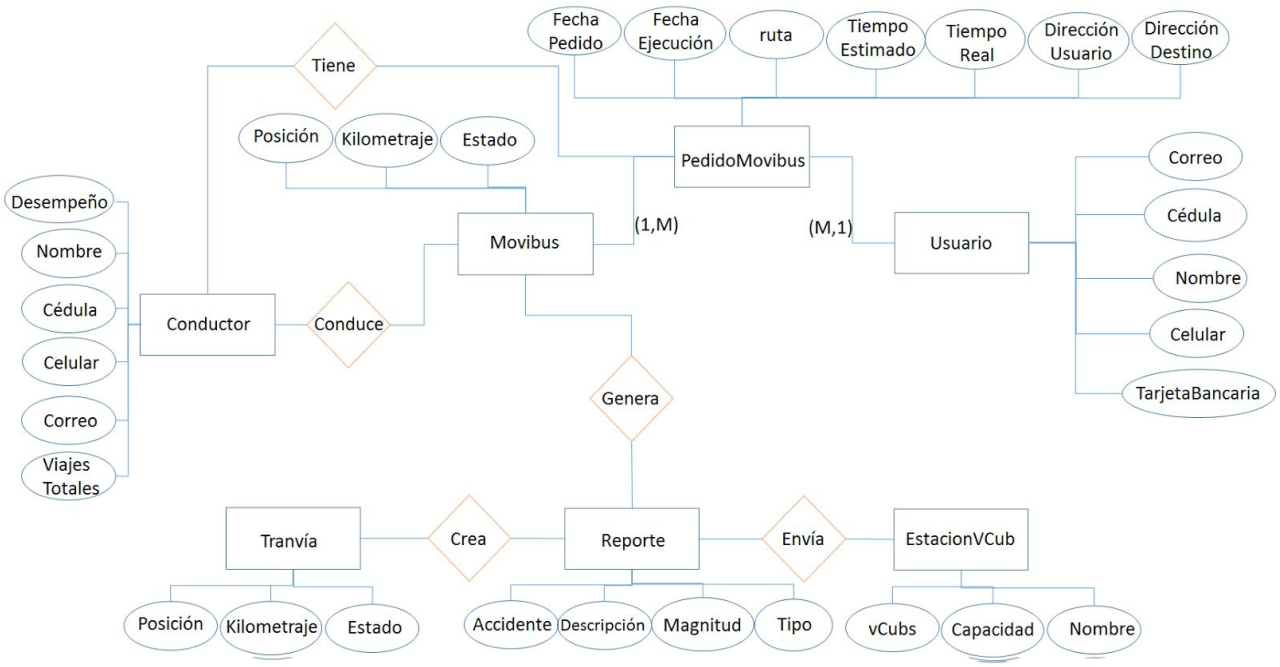


Diagrama de paquetes

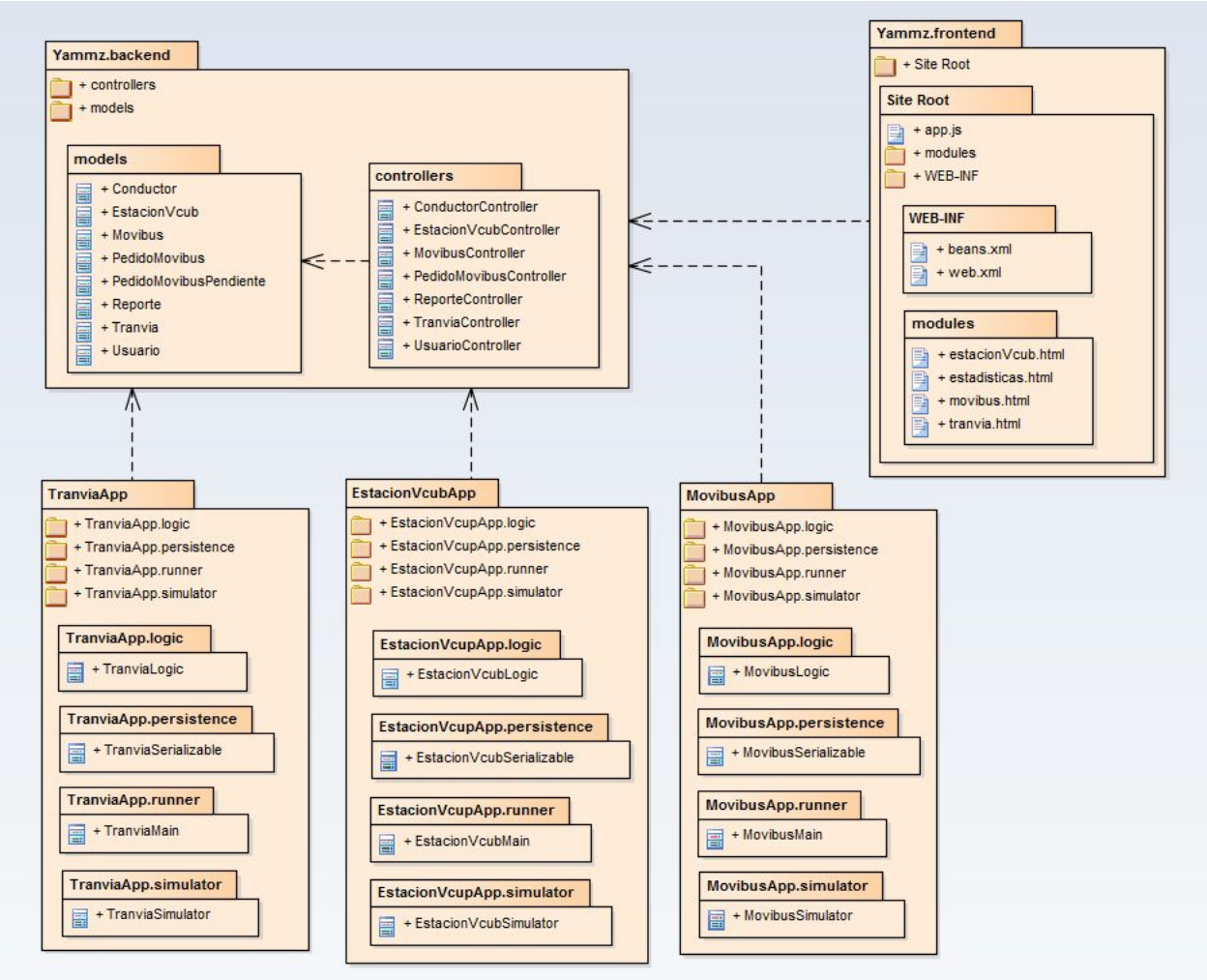
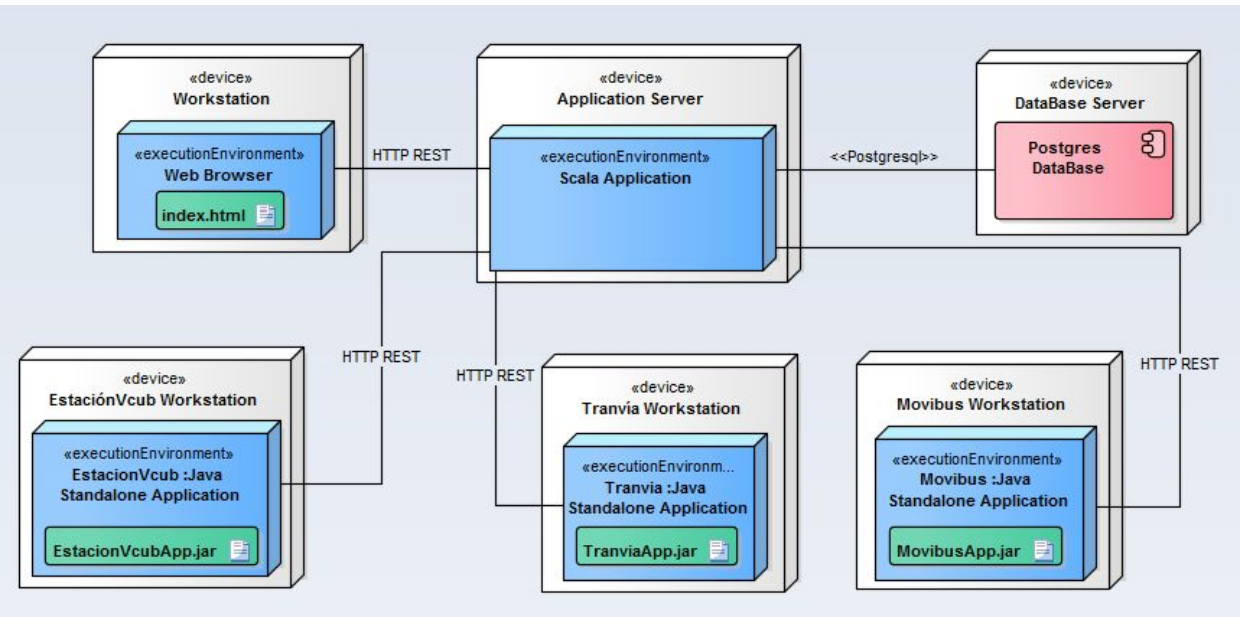


Diagrama de despliegue



7. Análisis de métricas

Después de implementar la base de datos en la aplicación, se han vuelto a ejecutar los requerimientos que ofrecen mayor reto a la aplicación y su arquitectura; a saber estos son aquellos relacionados con crear, obtener y modificar la información de los vehículos presentes en TBC. Se realizaron entonces pruebas de métodos POST, PUT y GET para 250 hilos de ejecución de movibuses y tranvías, y 400 hilos de estaciones de Vcubs cada una con una capacidad de 10 Vcubs, para obtener así el total de 4500 vehículos presentes.

Se tiene entonces a continuación los resultados obtenidos para todas las pruebas definidas

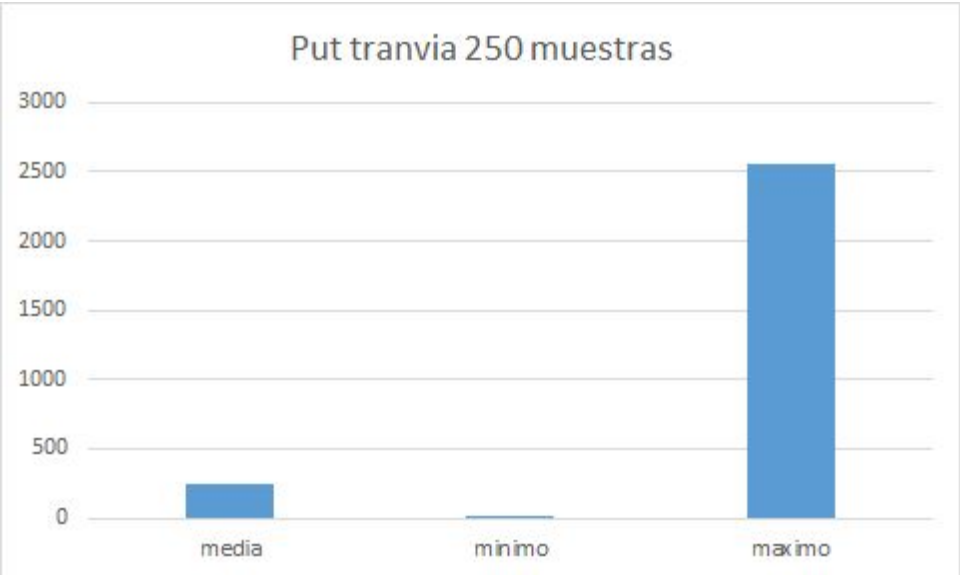
Todas las gráficas a continuación están en milésimas de segundo para el eje Y:

7.1. Get-Estación Vcub

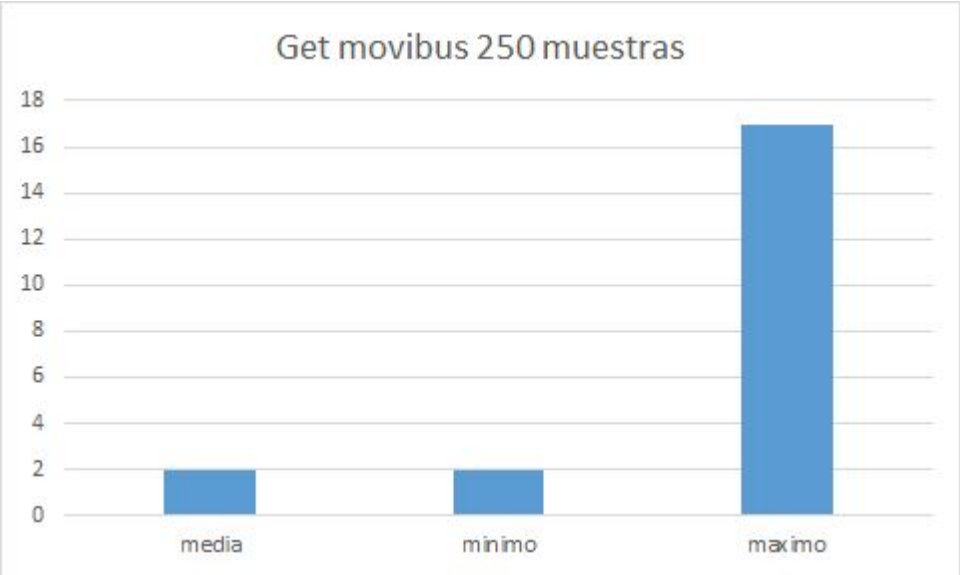
7.2. Post-Estación Vcub

7.3. Post-Tranvia

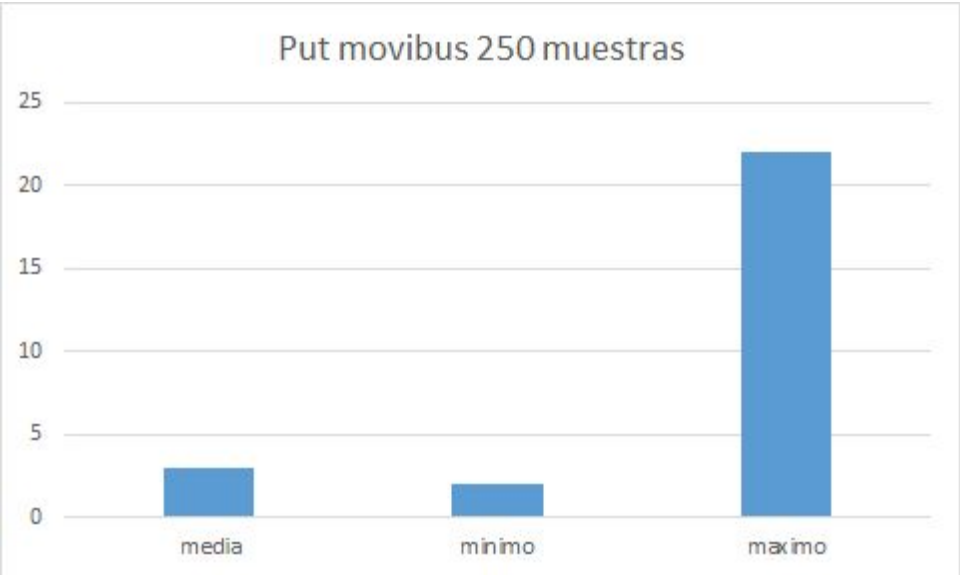
7.4. Put-Tranvia



7.5. Get-Movibus



7.6. Put-Movibus



De los resultados obtenidos resalta el buen desempeño de la aplicación. Este buen desempeño pudo lograrse gracias a la arquitectura por actores implementada en el framework de Play, puesto que ésta permite que la aplicación no pierda gran desempeño en cuanto aumenta la cantidad de pedidos hechas a esta (escalabilidad) . Destaca además el hecho de que la base de datos relacional no genera retrasos en la aplicación puesto que no son necesarias numerosas operaciones de Join entre las tablas la misma en el tiempo de ejecución al solicitar un requerimiento.

Se puede observar que el máximo de las gráficas es un valor muy elevado con respecto a los otros datos y esto es debido a que el servidor pasa de un estado de inactividad a uno en que ocurre un constante envío de requests, lo que causa que los primeros pedidos tomen un tiempo considerablemente mayor a los que ocurren después.

En general, los métodos con peor rendimiento son aquellos encargados de hacer PUT y POST. Esto es fácil de explicar si se tiene en cuenta que estos implican que Postgresql debe realizar operaciones tanto de lectura como de escritura. A pesar de que algunos de los métodos POST superaron el tiempo máximo exigido por el enunciado, estos no se consideran servicios de la aplicación, puesto que no es un requisito funcional crear 250 movibuses o tranvías en la base de datos del negocio en una sola tanda que no se repetirá constantemente, debido a las necesidades del negocio.

Podemos ver como los resultados, con respecto al desempeño y a la escalabilidad, son mejores en esta entrega que en la primera, esto seguramente se debe a que la base de datos puede hacer un manejo más óptimo de los datos que se le envían mediante elementos de virtualización que le permitan evitar uso indebido de la memoria y ahorrar tiempo, mientras que en la entrega anterior el manejo de los datos se hacía en memoria principal y posiblemente la cantidad de datos implicaba que ocurrieran fallos de página que aumentan el tiempo de respuesta a las peticiones.

El tiempo de respuesta de los servicios GET, permite a las aplicaciones StandAlone y a la aplicación web, recuperar rápidamente la información de estado y por tanto ofrecer la última información en todo momento incluso después de un fallo. El tiempo de respuesta rápido de los PUT permite a los vehículos mostrar su posición en los mapas de TBC cada cortos periodos de tiempo ~5-7 segundos por envío de posición; y a las estaciones de Vcubs a mantener los reportes de Vcubs al día en todo momento.

En conclusión, se lograron los requerimientos esperados bajo las métricas esperadas, el trabajo fue realizado por todos los integrantes de grupo, se trabajaron por separado las aplicaciones Standalone, el front-end y el back-end para después ser integradas en un proceso relativamente sencillo. Se intentó además hacer el despliegue cloud utilizando dos servicios PaaS diferentes, sin resultados satisfactorios

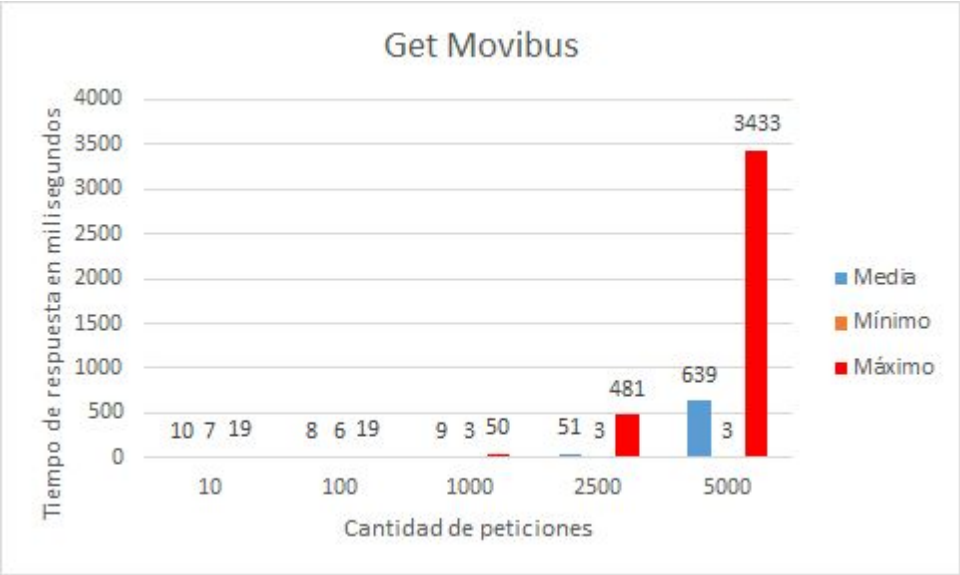
8. Resumen de métricas post-implementación de balanceador de carga

A continuación se presenta en tablas y gráficas los resultados obtenidos en JMeter al ejecutar cada uno de los requerimientos relevantes para TBC después de implementar el balanceador de carga para dos servidores destinados al balanceo.

8.1. Get Movibus

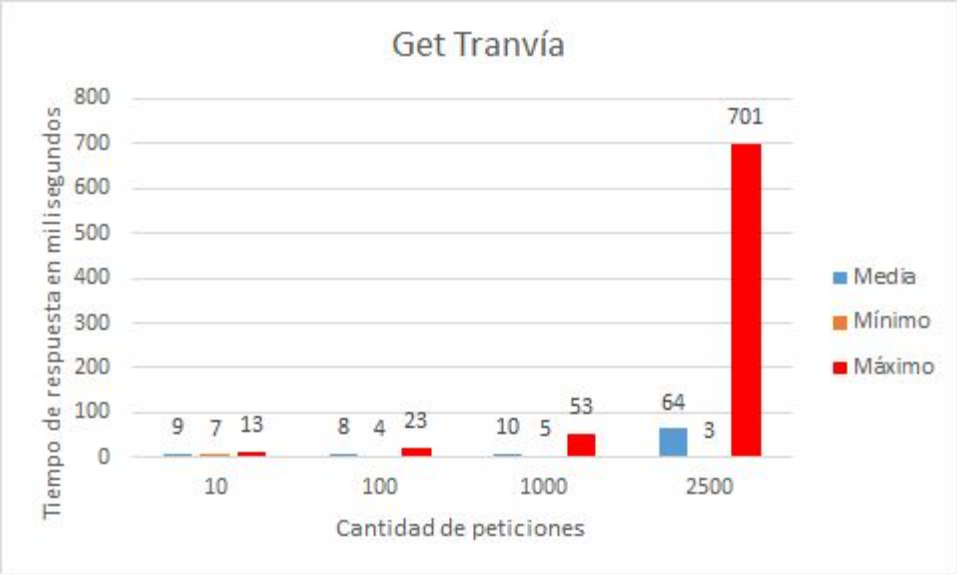
Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	10	7	19	3.7802116	0	2.2172949	0.452553	209
100	8	6	19	3.2404320	0	20.0360649	4.088413	208.95
1000	9	3	50	5.3406770	0	183.722212	37.49171	208.965
2500	51	3	481	100.10939	0	442.791357	90.37440	209

5000	639	3	3433	842.25914	0.1358	666.844492	276.7116	424.9158



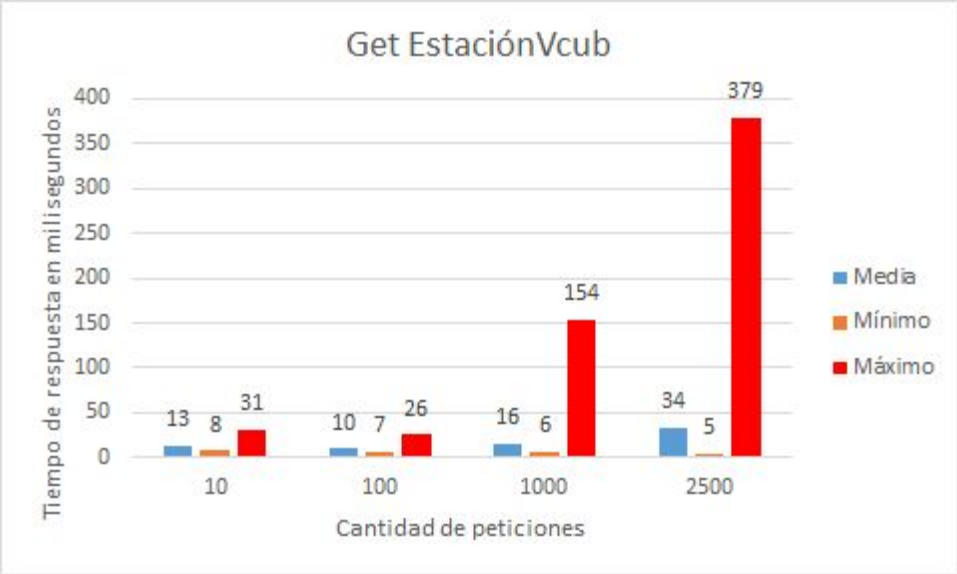
8.2. Get Tranvía

Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	9	7	13	2.0024984	0	2.2168033	0.4632772	214
100	8	4	23	3.3682488	0	20.032051	4.1854075	213.95
1000	10	5	53	6.1365084	0	184.46781	38.545485	213.97
2500	64	3	701	138.04346	0	435.00957	90.910203	214
5000	-							



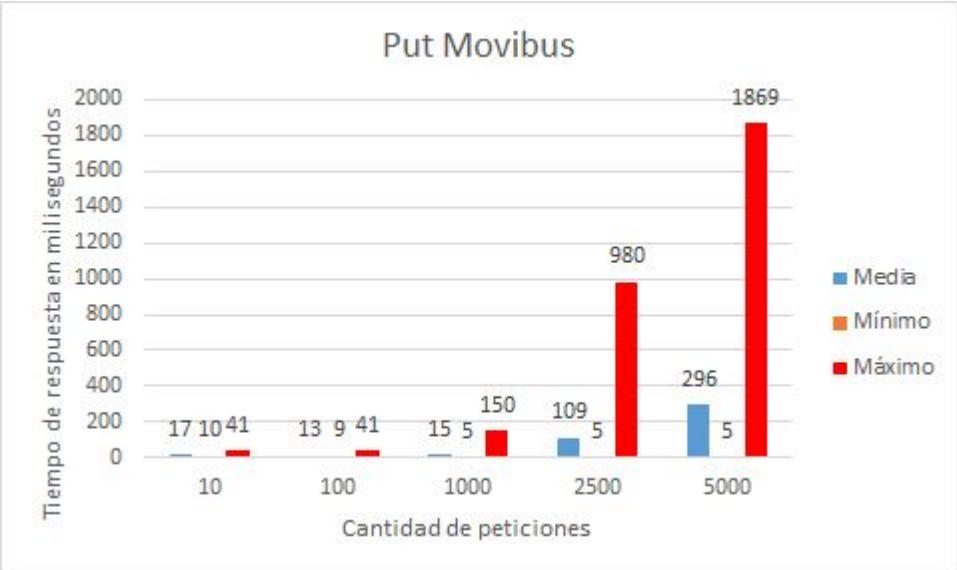
8.3. Get EstaciónVcub

Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	13	8	31	6.7852781	0	2.21582096	0.4327775	200
100	10	7	26	3.6728190	0	19.8373339	3.8744792	200
1000	16	6	154	13.388950	0	174.246384	34.032497	200
2500	34	5	379	49.999010	0	410.037723	80.085492	200
5000	-							



8.4. Put Movibus

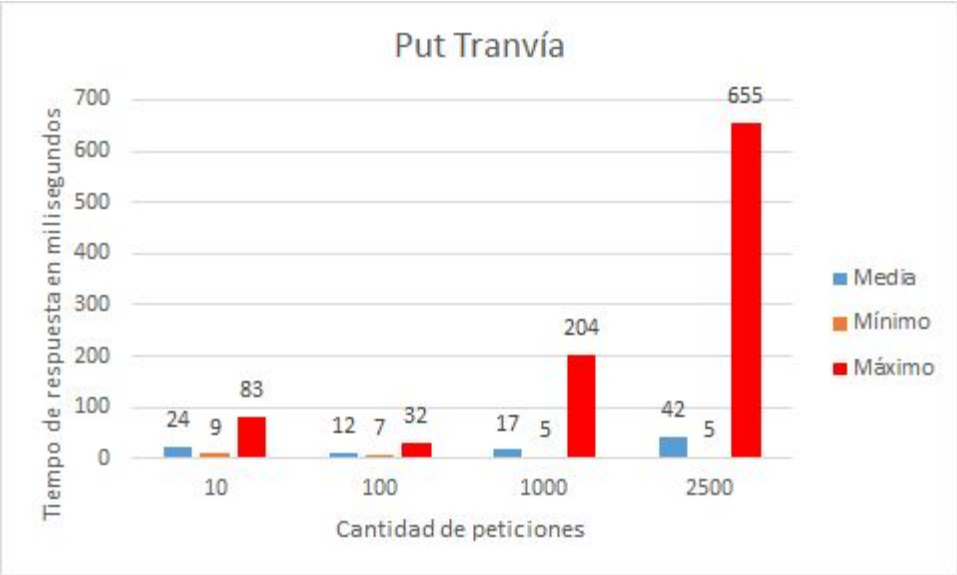
Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	17	10	41	9.5608577	0	2.21336875	0.4517520	209
100	13	9	41	5.1427132	0	19.8137507	4.0440174	209
1000	15	5	150	14.189349	0	190.585096	38.896854	208.99
2500	109	5	980	209.27836	0	446.907401	91.214498	209
5000	296	5	1869	456.72073	0.0444	699.496363	190.99256	279.596



8.5. Put Tranvía

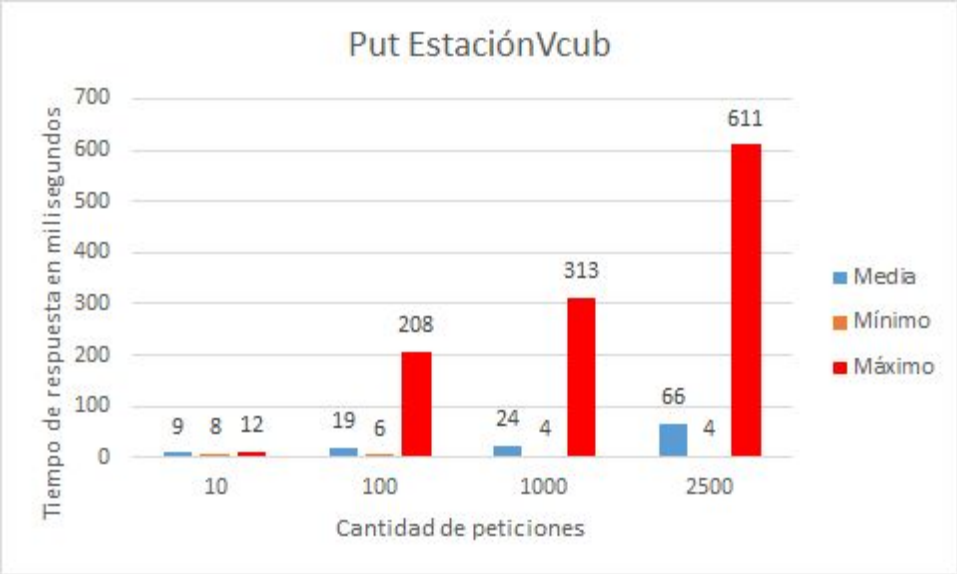
Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes

10	24	9	83	23.359214	0	0.0180901 1	0.0038383	217.27272
100	12	7	32	4.9385321	0	20.008003 2	4.1803830	213.95
1000	17	5	204	22.113334	0	181.65304 3	37.958212	213.975
2500	42	5	655	91.622279	0	445.79172 6	93.163505	214
5000	-							



8.6. Put EstaciónVcub

Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	9	8	12	1.2489996	0	2.21582096	0.4327775	200
100	19	6	208	38.538680	0	20.1045436	3.9266686	200
1000	24	4	313	26.310473	0	193.386192	37.770740	200
2500	66	4	611	96.877362	0	436.833828	85.319107	200
5000	-							



En general, se encontró que la latencia aumentó para la misma cantidad de peticiones en comparación con la entrega pasada. Este resultado es esperado al implementar el balanceador de carga, puesto que dado que el balanceador de carga agrega latencia al direccionar una petición; para pocas peticiones el balanceador no ayuda en desempeño, sólo mejora disponibilidad. Para un número mayor de peticiones se encontró que la latencia disminuye, y se encuentra que ahora se pueden atender un mayor número de peticiones que las que antes por petición. Por ejemplo, en la anterior entrega no se podían realizar 5000 peticiones de Put Movibus porque JMeter dejaba de responder, en cambio en esta entrega se logró responder e incluso la media del tiempo de respuesta estuvo dentro de la esperada.

Como fue mencionado antes, la implementación del balanceador de carga afectó un poco el desempeño, puesto que a cada petición le agrega un poco de latencia al tener que redireccionarla a un servidor, a cambio de obtener mejor escalabilidad y mayor disponibilidad. La mejora a la escalabilidad y disponibilidad se logra dado que al repartir las solicitudes entre servidores estos no se sobrecargan, lo que permite que más solicitudes sean atendidas sin fallas en la aplicación. Una mejora adicional a la disponibilidad viene del hecho que si un servidor se cae, el balanceador se encarga de redireccionar las peticiones a los demás servidores activos.

La configuración que se usó para el balanceador fue Round-Robin, debido a que esta es la mejor a elegir cuando se tienen servidores que son iguales o parecidos en especificaciones técnicas. Esto se debe a que Round-Robin hace que el balanceador rote las peticiones entre los servidores en orden: Al tener los servidores especificaciones iguales, dado que éstos son las máquinas virtuales, se puede asumir que toda petición es atendida en tiempos iguales y que al usar Round-Robin ningún servidor se sobrecarga y se ahorra tiempo al no tener que preguntar características como el servidor con menor cantidad de conexiones.

Debe destacarse que todos los resultados anteriores se obtuvieron balanceando la carga entre dos servidores. Se espera que al aumentar la cantidad de servidores disponibles aumente en buena medida la escalabilidad y disponibilidad del proyecto, puesto que se encontraría más difícil que un servidor se sobrecargue a tal punto que no sea capaz de responder a más solicitudes.