Software Architecture Documentation
SAD
Caso TBC

Grupo 04 YAMMZ
Carlos Felipe Agudelo Ospina
Samuel Baquero
Gabriel Martinez
Luis Miguel Mejía Nader
Sergio Yodeb Velásquez Yepes

Arquitectura y Diseño de Software
Gilberto Pedraza
Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
Bogotá D.C.
2015-2

Tabla de contenidos

1. **Descripción del documento**
   - 1.1. **Propósito y audiencia**
   - 1.2. **Organización del documento**
   - 1.3. **Términos y convenciones**
2. **Generalidades del proyecto**
   - 2.1. **Contexto y Problema a resolver**
   - 2.2. **Descripción del sistema**
   - 2.3. **Objetivos**
   - 2.4. **Grupos interesados (Stakeholders)**
3. **Contexto**
   - 3.1. **Actores**
   - 3.2. **Casos de uso**
4. **Requerimientos de calidad**
   - 4.1. **Árbol de calidad**
   - 4.2. **Escenarios de calidad**
5. **Puntos de vista**
   - 5.1. **Vista de contexto**
   - 5.2. **Vista funcional**
   - 5.3. **Vista de despliegue**
   - 5.4. **Vista de desarrollo**
   - 5.5. **Vista de información**
6. **Evaluación y experimentación**
   - 6.1. **Experimento 1**
   - 6.2. **Experimento 2**

1. Contexto

1.1. Propósito y audiencia

Se presenta en este trabajo la documentación de arquitectura de software desarrollada por el grupo Yammz dentro del contexto del proyecto de desarrollo e implementación de un sistema integrado de información para la empresa de transporte público TBC. Se encontrará en la documentación la información relevante de la arquitectura enfocada a los intereses de los *stakeholders* del proyecto, justificando las decisiones tomadas a partir del contexto del problema planteado.

1.2. Organización del documento

En el presente trabajo se desarrollará primero una breve descripción de la estructura y contenido de la documentación SAD, seguido por la descripción de las generalidades del proyecto en cuanto a el problema a resolver, los objetivos y los *stakeholders*. Posteriormente se expondrá la información del contexto que guió la toma de decisiones en el desarrollo de la arquitectura a un nivel más detallado para que el lector pueda entender posteriormente la justificación de las decisiones mencionadas. Después, se expondrán los requerimientos de calidad (requerimientos no funcionales) que se priorizaron al momento del diseño de la arquitectura. Continuando, se expondrán las vistas de la arquitectura más relevantes por stakeholder y para finalizar se mostrarán los resultados obtenidos después de experimentar sobre la arquitectura desarrollada.

1.3. Términos y convenciones

Se utilizará para la descripción de la mayoría de vistas y puntos de vista notación UML 2.

2. Generalidades del proyecto

2.1. Contexto y Problema a resolver

TBC es la empresa de transporte público de una cierta ciudad. Siguiendo una política de conservación del medio ambiente, ella ofrece tres tipos de medios de transporte: tranvía eléctrico, movibus y vcub.

El tranvía circula sobre carriles en las propias calles, sin separación del resto de la vía. Tres líneas de tranvía (líneas A, B y C) aseguran los ejes principales de la red y se cruzan en el centro de la ciudad. Las líneas circulan desde las 5 a.m. hasta medianoche.

Movibus es un bus destinado a transportar personas cuya movilidad es reducida (por ejemplo, personas en sillas de ruedas). Las reservaciones del servicio deben hacerse con una semana de anticipación. Si no es posible de satisfacer un servicio, la solicitud es colocada en lista de espera. Si un mobibus es liberado, el usuario será advertido. El transporte será definitivamente planificado con el acuerdo del usuario. Si la noche anterior al día de la reservación no hay ninguna posibilidad de transporte, el usuario es igualmente informado.

Vcub es una bicicleta disponible en estaciones ubicadas en ciertos puntos de la ciudad. El servicio es accesible con tarjetas bancarias. El usuario presta una bicicleta en una estación de su elección y la restituye en cualquiera de las otras según su recorrido.

TBC cuenta con 250 tranvías, 250 mobibus y 4000 vcubs.

Uno de los principales problemas que enfrenta actualmente Tbc es el control oportuno de sus vehículos. Para la empresa es vital poder conocer la posición actual de todos sus vehículos y el estado de cada uno de ellos.

En primer lugar, TBC desea conocer la posición de todos sus vehículos para tomar decisiones de negocio oportunas, por ejemplo:

- Ante una solicitud de mobibus poder enviar el vehículo disponible que se encuentre más cerca al usuario.
- Si una estación Vcub cuenta con solo 10% de bicicletas disponibles pedir al servicio encargado que la estación sea llenada al 100% la noche del incidente.

En segundo lugar poder controlar la ruta seguida por los conductores de movibus, los tiempos de cada trayecto.

En tercer lugar, se desea que en caso de un accidente de tranvía se pueda determinar la magnitud de la emergencia, y en caso de ser necesario enviar buses de reposición para cubrir el trayecto que el tranvía dejó de cubrir. TBC ha decidido instalar dispositivos GPS en cada uno de los movibuses.

Igualmente, para controlar el uso de vcubs, estas son aseguradas a un terminal interactivo, este terminal informa al sistema central si una bicicleta ha sido prestada o restituida.

Adicionalmente, los tranvías cuentan con sensores de choque, temperatura y botón de pánico para determinar posible situaciones de emergencia. Esta información es enviada al sistema central para que esta envíe los buses de reposición si es necesario.

TBC desea poder contar con centro de monitoreo y control en la ciudad. En este centro se reciben las señales provenientes de todos los vehículos y se presentan en diferentes pantallas, indicando en un mapa la ubicación de cada vehículo (tranvía y movibus) en un momento dado así que la disponibilidad de bicicletas en cada estación. El sistema central debe permitir realizar consultas sobre los vehículos tranvía y mobibus), kilometrajes transcurridos entre revisiones mecánicas, información de los conductores. Debe ser posible generar reportes para controlar tiempos promedio, conductores más efectivos y menos efectivos en trayectos, tipos de trayectos con más problemas, accidentes más comunes y en general todos aquellos indicadores que permitan mejorar la operación de TBC.

Es vital que la información provenientes de los vehículos sea transmitida, recibida y procesada de forma inmediata. La seguridad de la información y su confidencialidad es vital, por lo que los datos transmitidos y almacenados no pueden ser interpretados ni manipulados por personas no autorizadas. El sistema debe ser altamente

disponible dado que los vehículos transportan usuarios los 365 días del año, las 24 horas del día.

2.2. Descripción del sistema

El sistema a implementar en TBC debe de contar con gran desempeño y tiempos de respuesta cortos dada la naturaleza de sus operaciones. También debe contar con buena escalabilidad para soportar la cantidad de vehículos actual y permitirle a la empresa expandirse. Otros atributos de calidad a considerar son la seguridad de la información que viaja dentro del sistema de comunicación de TBC y la mantenibilidad del software desarrollado.

Expandiendo sobre las características del sistema a implementar en la empresa TBC, se identifican ciertos componentes que no han de cambiar sin importar la arquitectura a implementar. Dichos componentes son:

- **Aplicación standalone para movibus:** Software que puede encontrarse dentro de cada movibus de la empresa para permitirle a éste comunicarse con el servidor central de TBC, recibir y transmitir su información.
- **Aplicación standalone para estación vcub:** Este componente permite a una estación de vcubs comunicar al sistema central su estado y hacer solicitud de vcubs en caso de que necesite.
- **Aplicación standalone para tranvía:** Brinda a un tranvía la posibilidad de reportar su estado y reportar una emergencia al servidor central de la empresa.
- **Aplicación de control:** Debe tenerse un conjunto de componentes de software que actúe como centro de control para cada uno de los tres componentes antes descritos y además brinde funcionalidades a los usuarios y administradores del sistema. Este componente es el más susceptible a cambios puesto que es el componente central del sistema y es aquel que brinda las funcionalidades requeridas del sistema.

Se puede especificar también la información que viaja entre los componentes del sistema:

- **App. Movibus-App. Central:** Información viaja entre ambas aplicaciones. El sistema central envía al componente de movibus información de pedidos y rutas mientras que la aplicación de movibus envía al componente central información de su estado.
- **App. Estación Movibus-App. Central:** Información viaja únicamente del componente de estación movibus a la aplicación central, informando su estado y la necesidad de vcubs si existe.
- **App. Tranvía-App. Central:** La aplicación de tranvía es la que envía información al componente central, enviando información sobre el estado del tranvía y reportando cualquier posible accidente.

2.3. Objetivos

Se espera desarrollar un sistema de comunicación e información para la empresa de transporte público TBC que permita a los componentes mencionados en el numeral 2.2. desarrollar sus funciones teniendo en cuenta los siguientes requerimientos planteados por los stakeholders y tomados como objetivos del trabajo desarrollado:

- Responder todos los requerimientos del sistema en menos de 1 segundo.
- Atender las solicitudes de 4500 vehículos concurrentes en un periodo de 5 segundos.
- Brindar una disponibilidad del 99.95%.
- Asegurar la información que viaja entre los componentes del sistema.
- Asegurar la integridad de los datos que viajan entre los componentes del sistema.
- Permitir gran nivel de modificabilidad en los componentes del sistema.

2.4. Grupos interesados (Stakeholders)

- **Desarrolladores y grupo de evaluadores técnicos:** Este grupo se encuentra interesado más que todo en los aspectos técnicos que permiten a la arquitectura alcanzar los requerimientos de calidad que se priorizan dentro de esta.
- **Usuarios:** Interesados en los aspectos funcionales de la aplicación. Específicamente están interesados en pedir movibuses y vcubs de forma fácil, rápida y segura.
- **Accionistas:** Interesados en tener una plataforma confiable que sostenga la cantidad de vehículos con los que cuenta TBC y brinde a sus usuarios una buena experiencia de uso.
- **Empleados:** Interesados en usar una plataforma que les brinde suficiente información sobre los elementos de la empresa y les permita comunicarse a la central en poco tiempo y con facilidad.

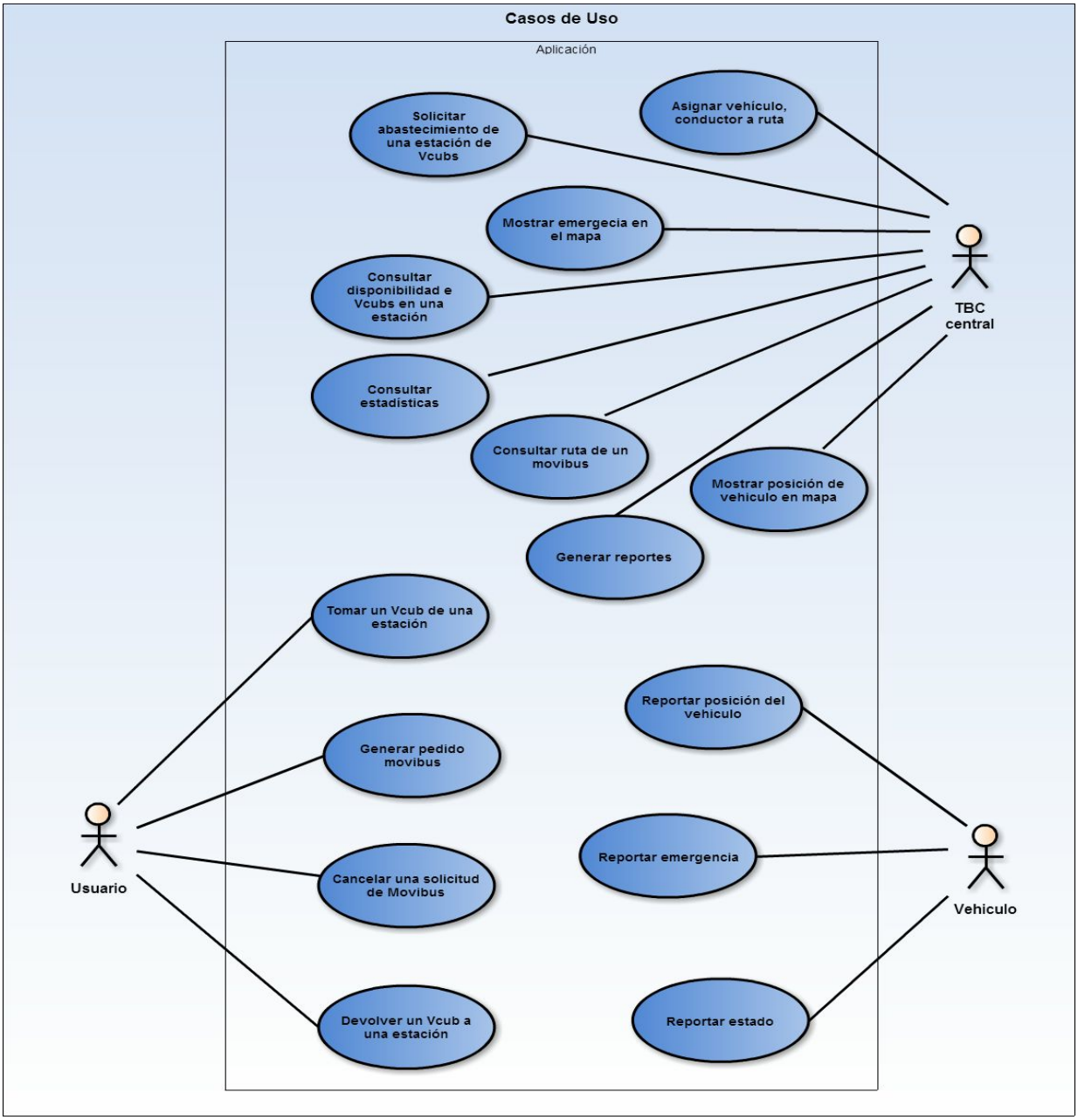
3. Contexto

3.1. Actores

A continuación se muestran los actores identificados dentro del contexto de TBC, describiendo su rol dentro del sistema:

- **Usuario:** Representa a un cliente de TBC. Su rol dentro del sistema es realizar pedidos de movibuses, realizar pedidos de vcbus en las estaciones y devolver vcubs a las mismas.
- **Vehículo:** Actor inanimado que representa a un vehículo que se comunica con el servidor central al enviar su información de estado y sus emergencias.
- **Central de TBC:** Representa la central de operaciones de TBC, encargada de recibir información de los vehículos y atender solicitudes de los clientes.

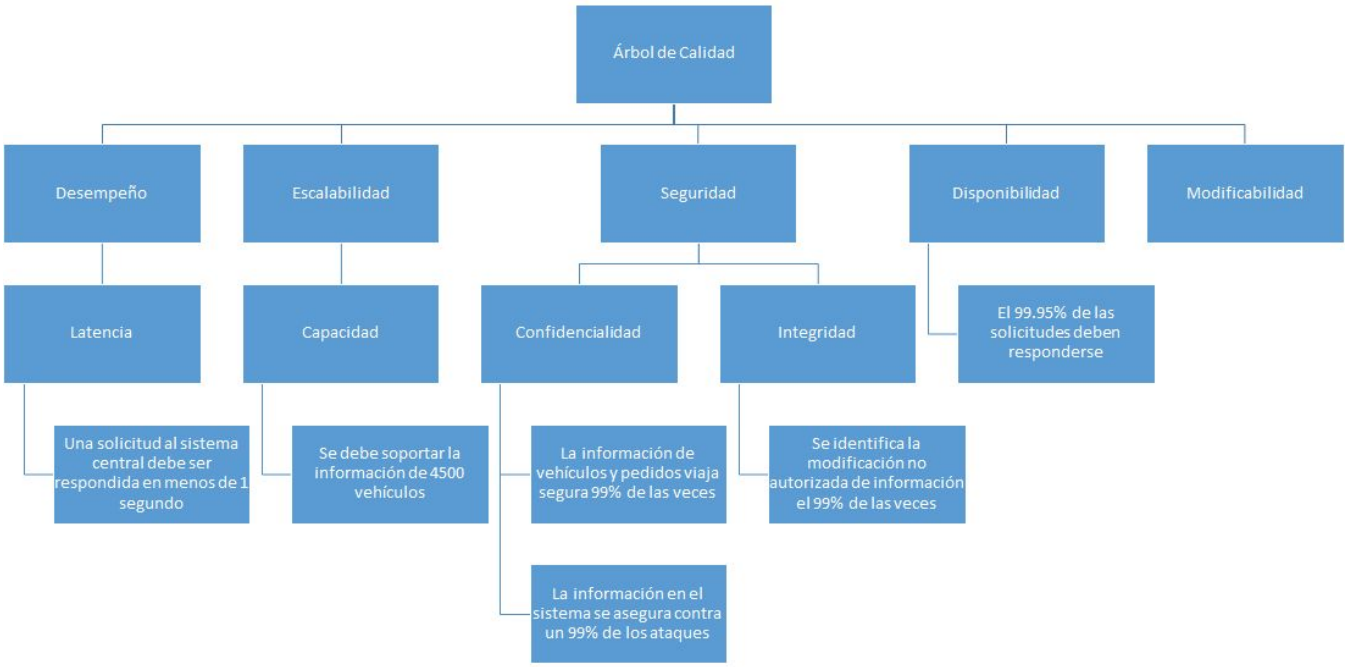
3.2. Casos de uso



4. Requerimientos de calidad

4.1. Árbol de calidad

En el siguiente árbol de calidad se presentan los atributos de calidad más importantes (nivel 1), con los subatributos que se tendrán en cuenta para cada atributo de calidad (nivel 2) y por último los escenarios de calidad a considerar para cada atributo (nivel 3).



4.2. Escenarios de calidad

A continuación se muestran los escenarios de calidad planteados para el proyecto clasificados a partir del atributo de calidad al que responden:

- **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.		

- **Disponibilidad:**

Identificador	**Tipo**	**Prioridad**
EC17	Disponibilidad	Alta
Fuente		
Usuario		
Estímulo		
Solicitar un movibus		
Ambiente		
Normal		
Medida esperada		
Se espera que el 99.95 de solicitudes de movibuses sean atendidas por TBC		

Identificador	**Tipo**	**Prioridad**
EC18	Disponibilidad	Alta

Fuente
Usuario
Estímulo
Solicitar un vcub
Ambiente
Normal
Medida esperada
Se espera que el 99.95 de solicitudes de retiro de vcubs sean atendidas por TBC

Identificador	**Tipo**	**Prioridad**
EC19	Disponibilidad	Alta
Fuente		
Vehículos		
Estímulo		
Envío de información de estado		
Ambiente		
Normal		
Medida esperada		
Se espera que el 99.95 de envío de información de estado de vehículos sean atendidas por TBC		

Identificador	**Tipo**	**Prioridad**
EC20	Disponibilidad	Alta
Fuente		
Vehículos		
Estímulo		
Reporte de emergencia.		

Ambiente
Normal
Medida esperada
Se espera que el 99.95 de reportes de emergencia de vehículos sean atendidas por TBC

- **Seguridad**

Identificador	**Tipo**	**Prioridad**
EC21	Seguridad	Media
Fuente		
Usuario		
Estímulo		
Información de usuario		
Ambiente		
Normal		
Medida esperada		
Se espera mantener seguro la información de usuarios contra 99% de los ataques externos		

Identificador	**Tipo**	**Prioridad**
EC22	Seguridad	Media
Fuente		
Vehículos		
Estímulo		
Información de vehículos		
Ambiente		
Normal		
Medida esperada		

Se espera mantener seguro la información de vehículos contra 99% de los ataques externos

Identificador	**Tipo**	**Prioridad**
EC23	Seguridad	Media
Fuente		
Vehículos		
Estímulo		
Información de vehículos		
Ambiente		
Normal		
Medida esperada		
El 100% de los datos transmitidos de los buses al servidor central no han sido modificados		

Identificador	**Tipo**	**Prioridad**
EC24	Seguridad	Alta
Fuente		
Usuarios		
Estímulo		
Información		
Ambiente		
Normal		
Medida esperada		
Los usuarios tienen acceso a la información que se les permite.		

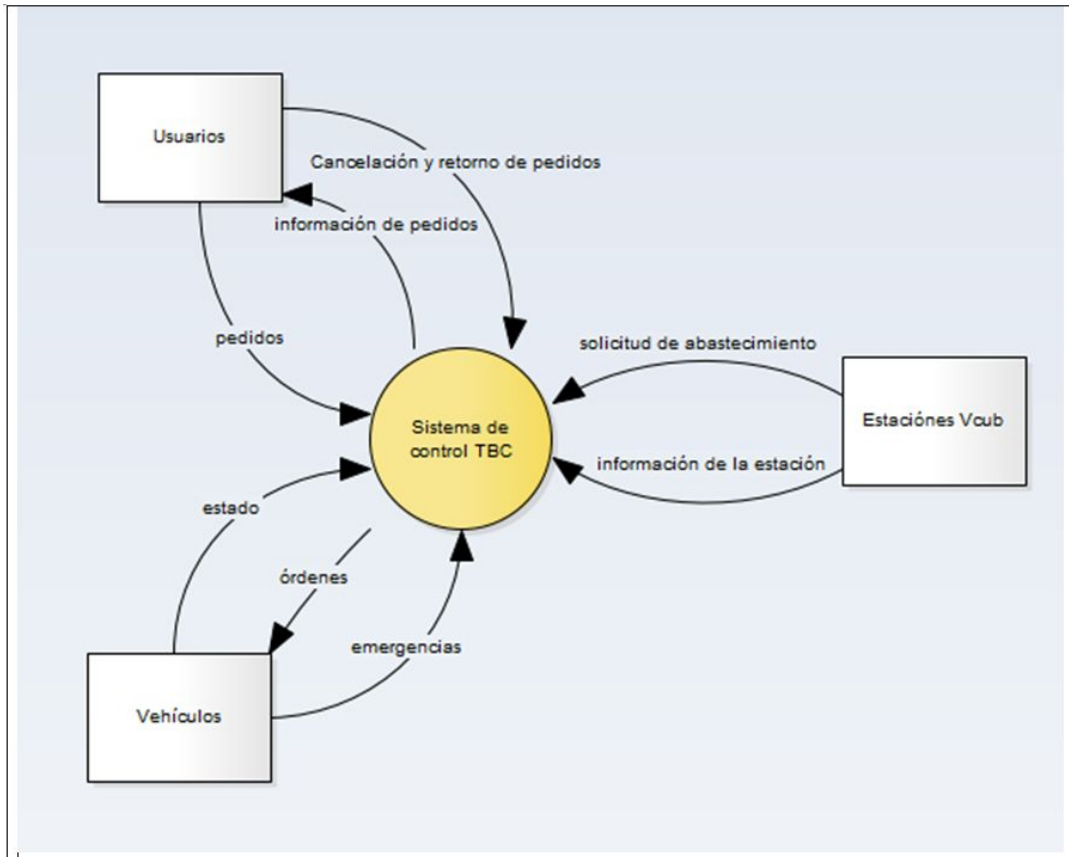
5. Puntos de vista

5.1. Vista de contexto

La vista de contexto está dirigida para todos los stakeholders de TBC. A través de ella se expresan las transacciones y solicitudes que el sistema debe soportar de forma general, mostrando a grandes rasgos la información que fluye entre los actores y componentes del sistema.

La vista de contexto está además complementada por el diagrama de casos de uso mostrado en el literal 3.2.

5.1.1. Diagrama de contexto



Familia
Module ( )
C&C ( )
Allocation ( )

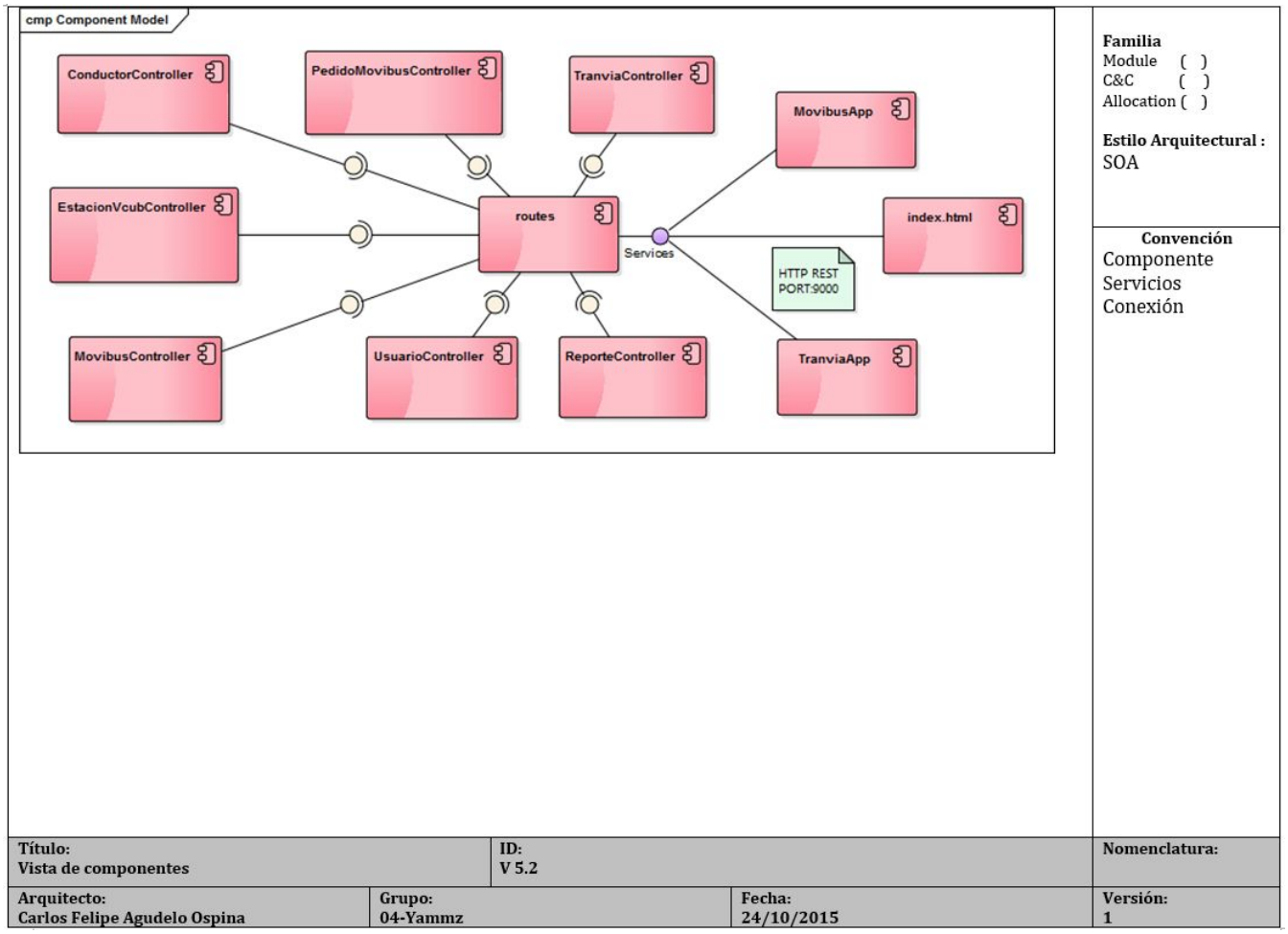
Estilo Arquitectural :
N/A

Convención
Rectángulo:
Actor
Circulo:
Sistema

Título: Vista de contexto	ID: V 5.1		Nomenclatura:
Arquitecto: Carlos Felipe Agudelo Ospina	Grupo: 04-Yammz	Fecha: 24/10/2015	Versión: 1

5.2. Vista funcional

5.2.1. Diagrama de componentes

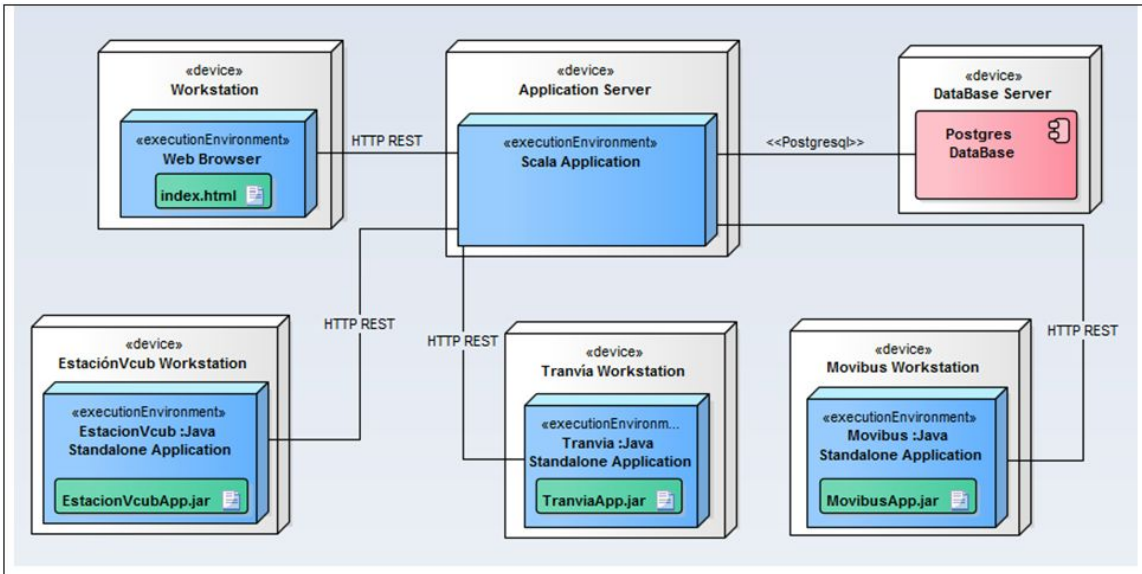


El diagrama de componentes expresa que los distintos controladores de la aplicación brindan sus servicios a un artefacto llamado routes dentro del proyecto principal. Es el artefacto routes el que enruta las distintas solicitudes HTTP REST hechas al puerto 9000 de los usuarios del sistema para que sean atendidas por los servicios correspondientes. Cada controlador se encarga de la lógica y de relegar la persistencia a la base de datos.

Se tiene entonces claro que la arquitectura es una orientada a servicios (SOA). Se tomó la decisión de optar por este estilo arquitectural por la modificabilidad que brinda, puesto que no importa cambios en la implementación de la lógica mientras que los servicios sean atendidos de igual forma, y por conocimiento y experiencia de este estilo.

5.3. Vista de despliegue

5.3.1. Diagrama de despliegue



Familia
Module ( )
C&C ( )
Allocation ( )

Estilo Arquitectural :
SOA

Convención
Nodo
Ambiente de ejecución
Componente
Artefacto

Título: Vista de despliegue	**ID:** V 5.3		**Nomenclatura:**
Arquitecto: Carlos Felipe Agudelo Ospina	**Grupo:** 04-Yammz	**Fecha:** 24/10/2015	**Versión:** 1

El diagrama de despliegue muestra que la aplicación se desplegará en un device llamado Application Server, encargado únicamente de mantener el programa en ejecución y listo para atender las solicitudes que lleguen de los distintos dispositivos externos, a saber:

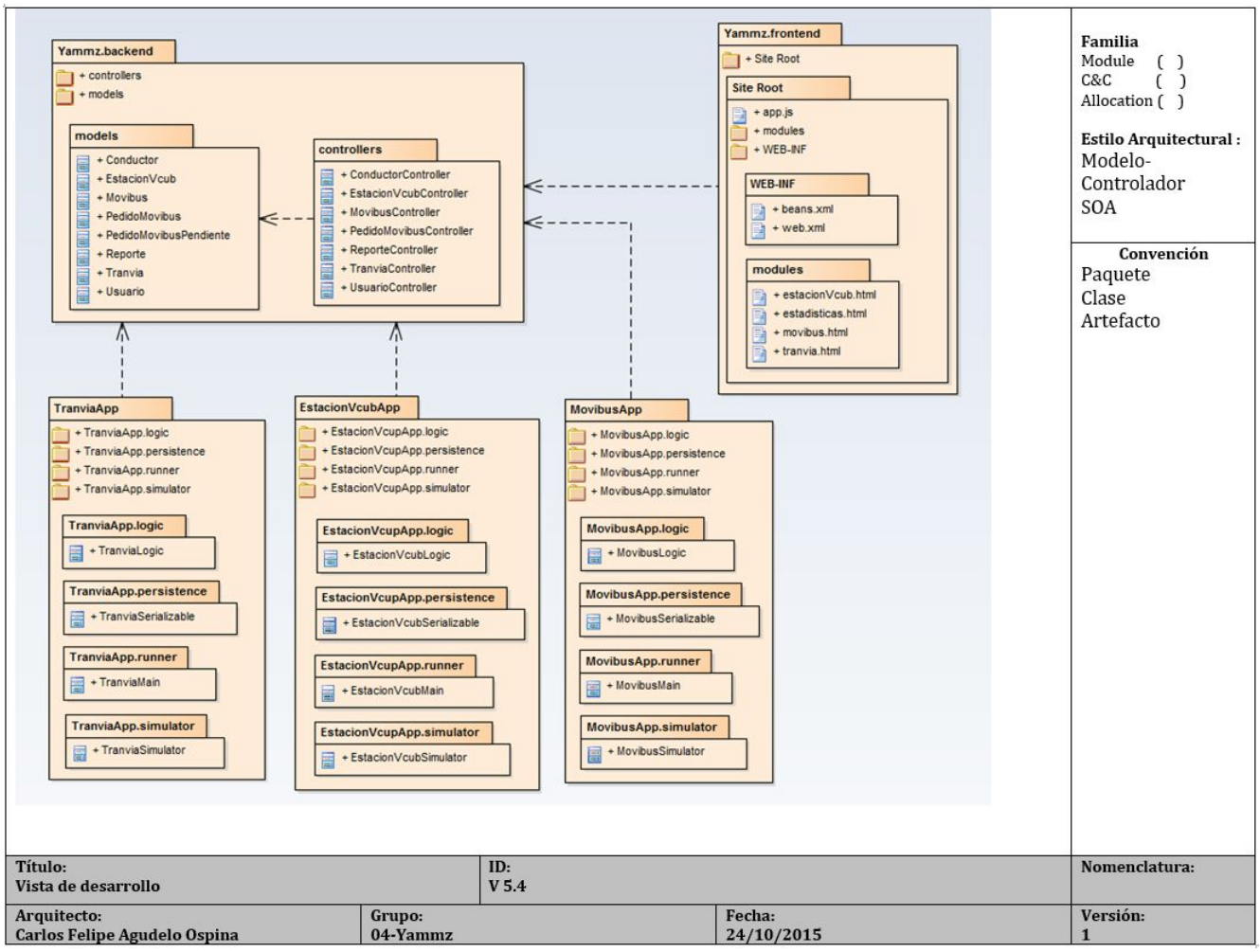
- Client Workstation: Representa al dispositivo desde el cual un usuario accede a los servicios de TBC, inicialmente este dispositivo deberá ser un ordenador dado que no se cuenta hasta el momento con soporte de aplicación móvil. Se tiene además que el ambiente de ejecución será el respectivo web browser del cliente que se encargará de desplegar el index.html desarrollado.
- Estación Vcub Workstation: Representa a un dispositivo dentro de una estación de vcubs que a partir de una aplicación standalone en Java podrá realizar solicitudes HTTP REST a la aplicación.
- Tranvía Workstation: Representa a un dispositivo dentro de un tranvía que a partir de una aplicación standalone en Java podrá realizar solicitudes HTTP REST a la aplicación.
- Movibus Workstation: Representa a un dispositivo dentro de un movibus que a partir de una aplicación standalone en Java podrá realizar solicitudes HTTP REST a la aplicación.

El último dispositivo relevante dentro del diagrama es el servidor de la base de datos, un computador encargado de manejar la base de datos de forma tal que el servidor de aplicación pueda guardar en él toda la información. Se decide tomar un dispositivo diferente

para no sobrecargar de más a la máquina del servidor de aplicación. El servidor de aplicaciones corre además nginx, de forma tal que se pueden distribuir las solicitudes entre otros servidores adicionales aumentando la escalabilidad y disponibilidad del servicio de TBC

5.4. Vista de desarrollo

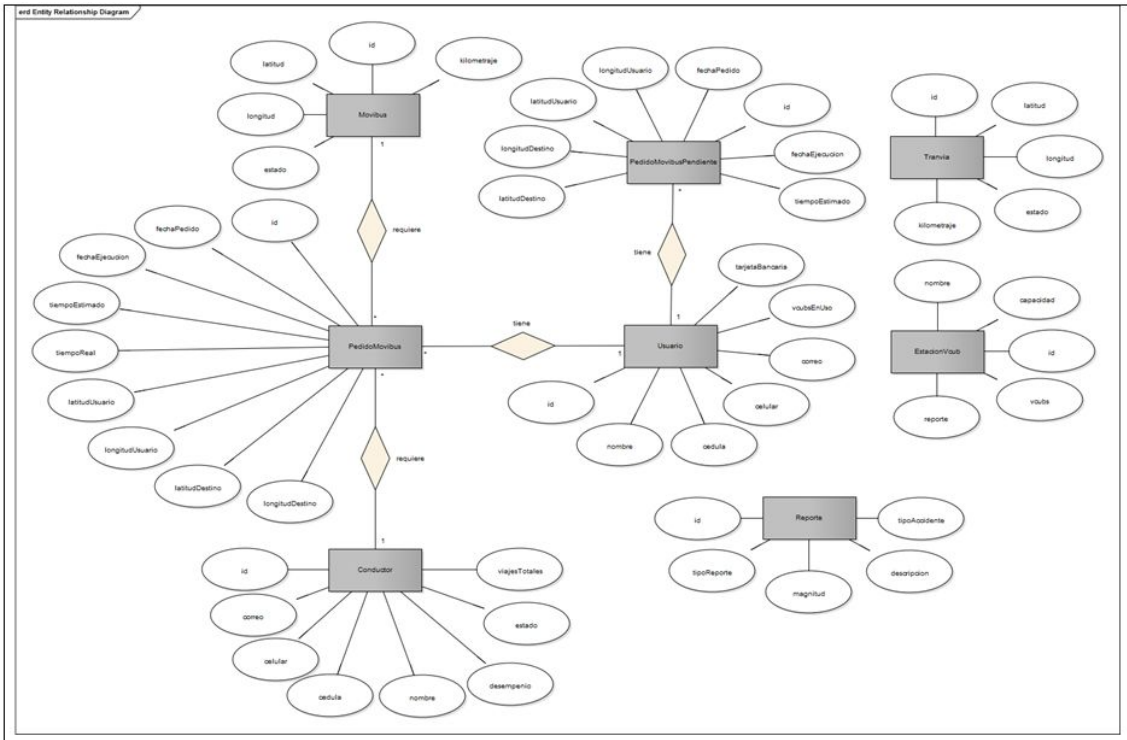
5.4.1. Diagrama de paquetes



Esta vista está destinada a los desarrolladores y expresa la organización en paquetes con las que cuenta el sistema. Este diagrama expresa además que el backend de la aplicación tiene un estilo Modelo-Controlador, implementado dado que se utiliza el framework Play! en este. Play! fue elegido como framework para el backend dada la escalabilidad que ofrece con la creación de actores en función de la demanda de servicios.

5.5. Vista de información

5.5.1. Diagrama de entidad-relación



Familia
Module ( )
C&C ( )
Allocation ( )

Estilo Arquitectural :
N/A

Convención
Rectángulo: Entidad.
Elipse: Atributo.
Rombo: Relación.

Título: Vista de información	ID: V 5.5		Nomenclatura:
Arquitecto: Carlos Felipe Agudelo Ospina	**Grupo:** 04-Yammz	**Fecha:** 24/10/2015	**Versión:** 1

6. Evaluación y experimentación

6.1. Experimento 1

El primer experimento realizado trató de verificar los requerimientos de disponibilidad y escalabilidad planteados por el problema al implementar todo el sistema con una base de datos.

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.

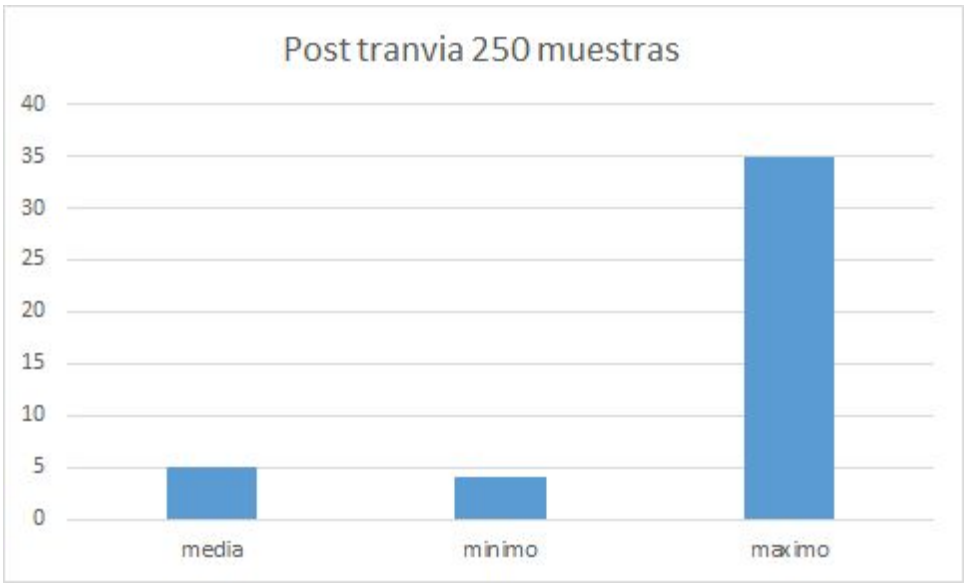
Se tiene entonces a continuación los resultados obtenidos para todas las pruebas definidas

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

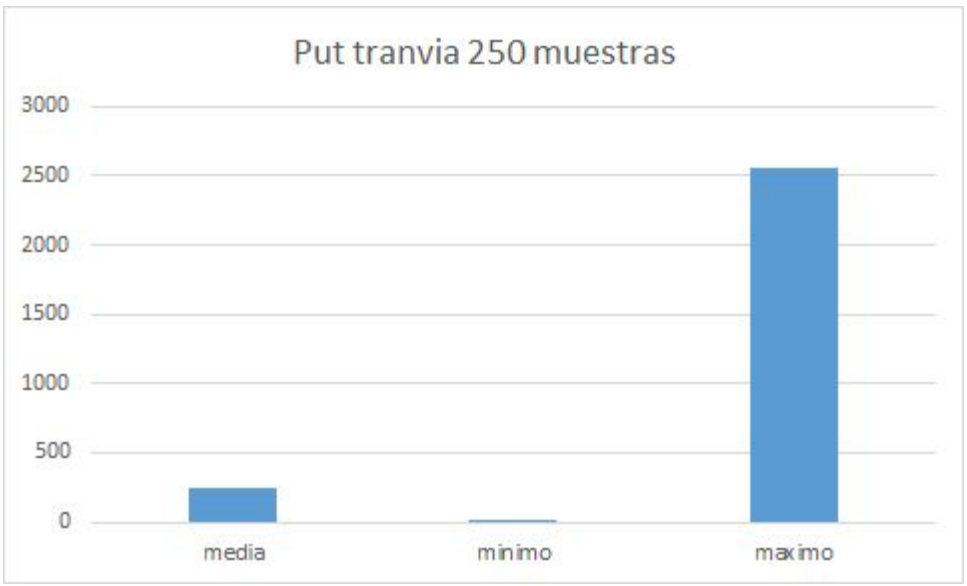
Get-Estación Vcub

Post-Estación Vcub

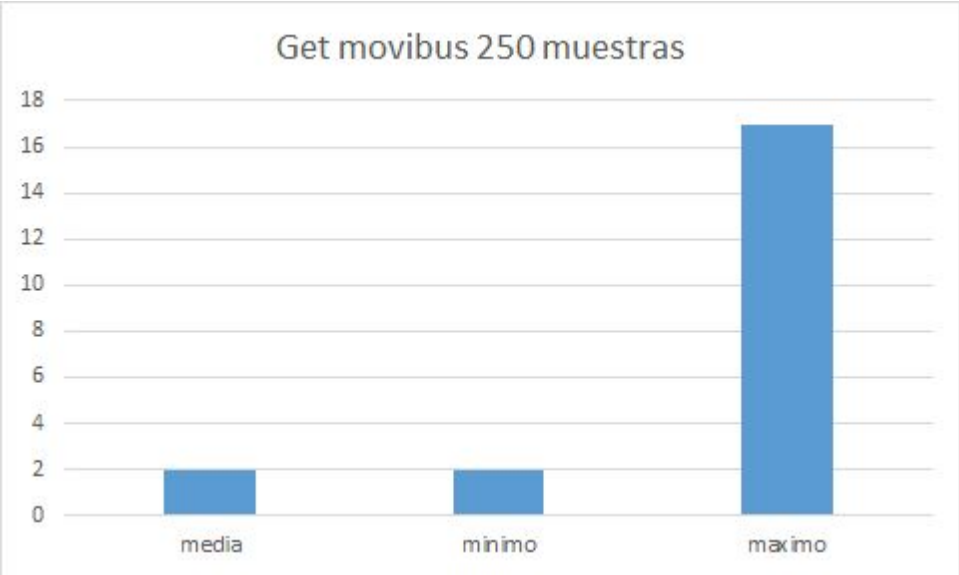
Post-Tranvia


Post tranvia 250 muestras
40
35
30
25
20
15
10
5
0
media
minimo
maximo

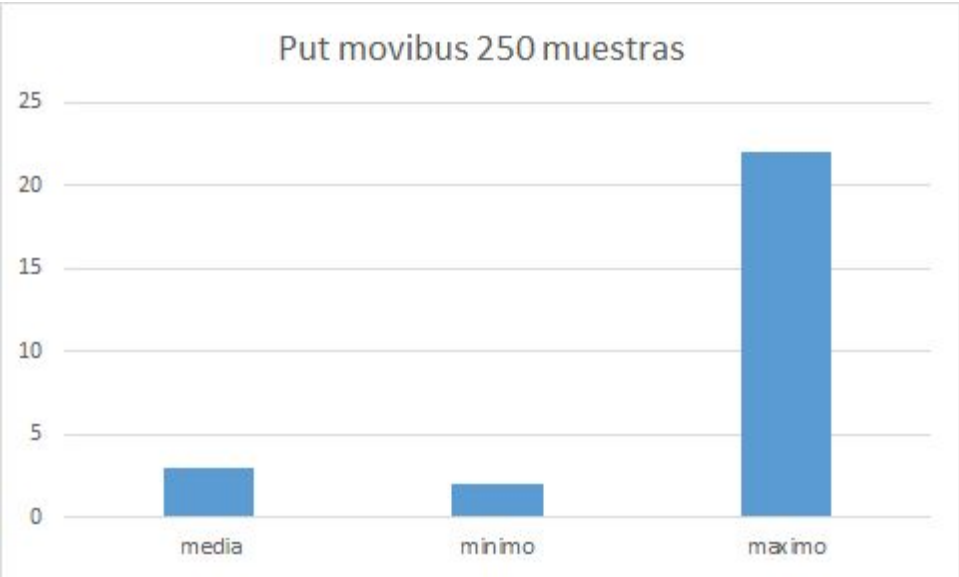
Put-Tranvia


Put tranvia 250 muestras
3000
2500
2000
1500
1000
500
0
media
minimo
maximo

Get-Movibus



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.

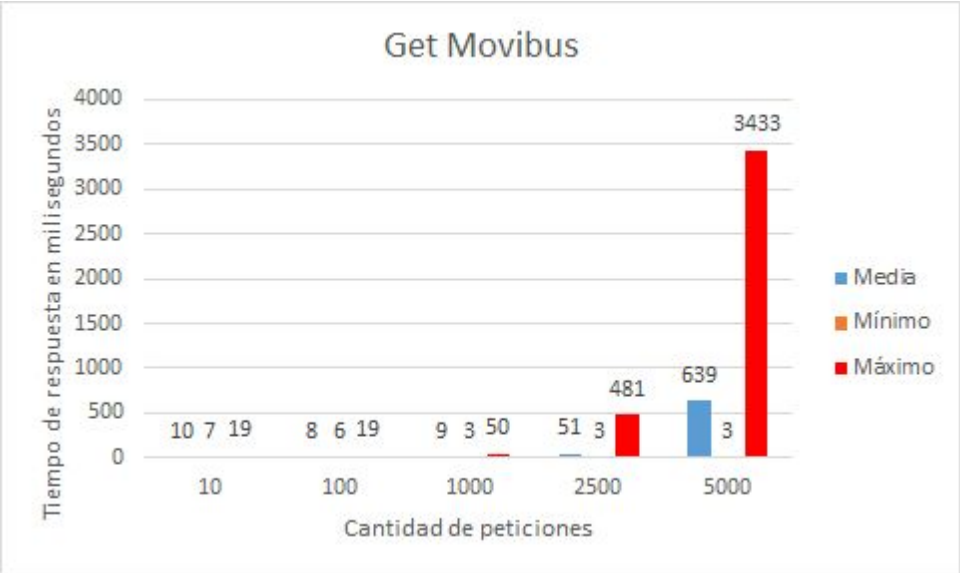
En conclusión, se lograron los requerimientos esperados bajo las métricas esperadas, el trabajo fue realizado por todos los integrantes de grupo, se trabajaron por separado las aplicaciones Standalone, el front-end y el back-end para después ser integradas en un proceso relativamente sencillo. Se intentó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

6.2. Experimento 2

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.

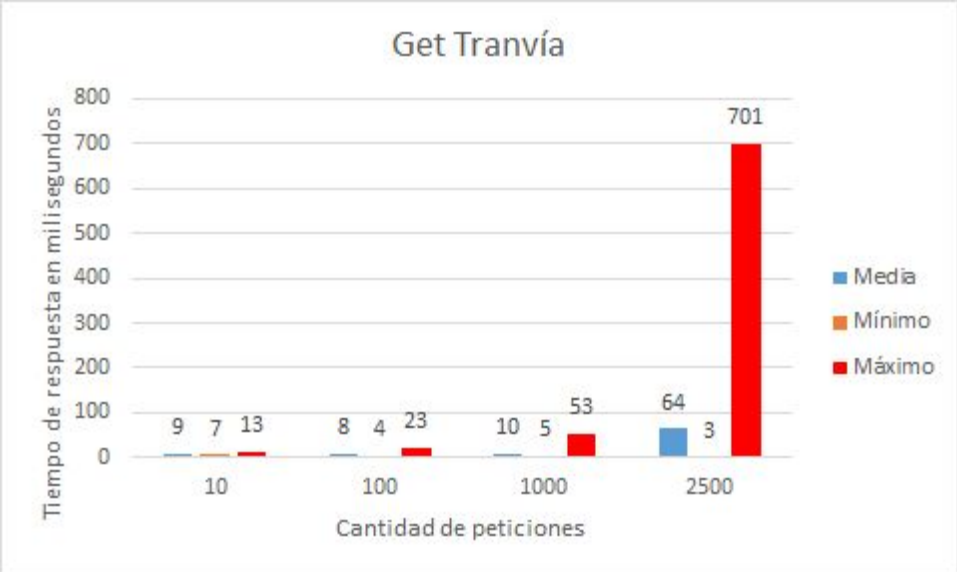
Get Movibus

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	20.036064 9	4.088413	208.95
1000	9	3	50	5.3406770	0	183.72221 2	37.49171	208.965
2500	51	3	481	100.10939	0	442.79135 7	90.37440	209
5000	639	3	3433	842.25914	0.1358	666.84449 2	276.7116	424.915 8



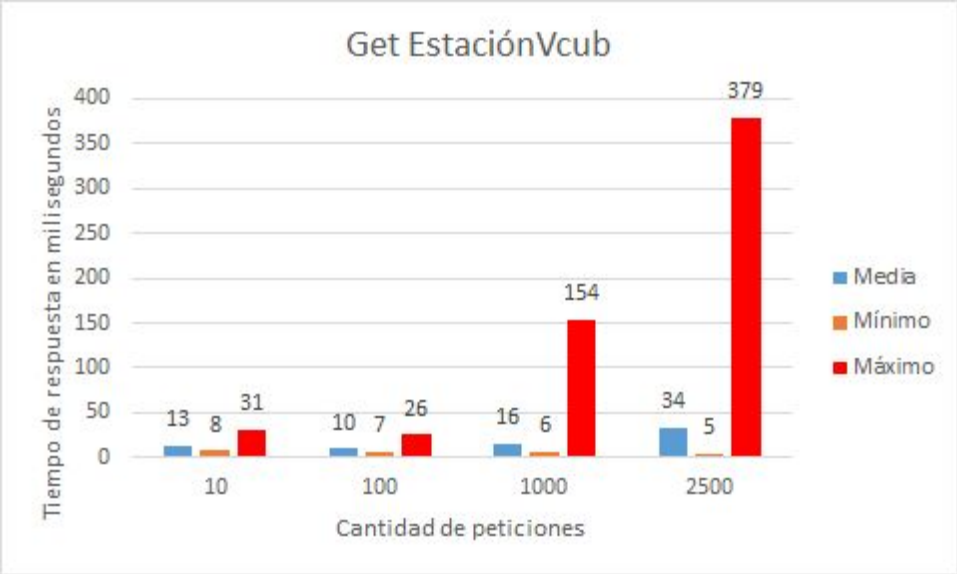
Get Tranvía

Samples	Avg	Mínimo	Máximo	Desv.Est.	% Error	Throughput	KB/sec	Bytes
10	9	7	13	2.0024984	0	2.2168033	0.463277 2	214
100	8	4	23	3.3682488	0	20.032051	4.185407 5	213.95
1000	10	5	53	6.1365084	0	184.46781	38.54548 5	213.97
2500	64	3	701	138.04346	0	435.00957	90.91020 3	214
5000	-							



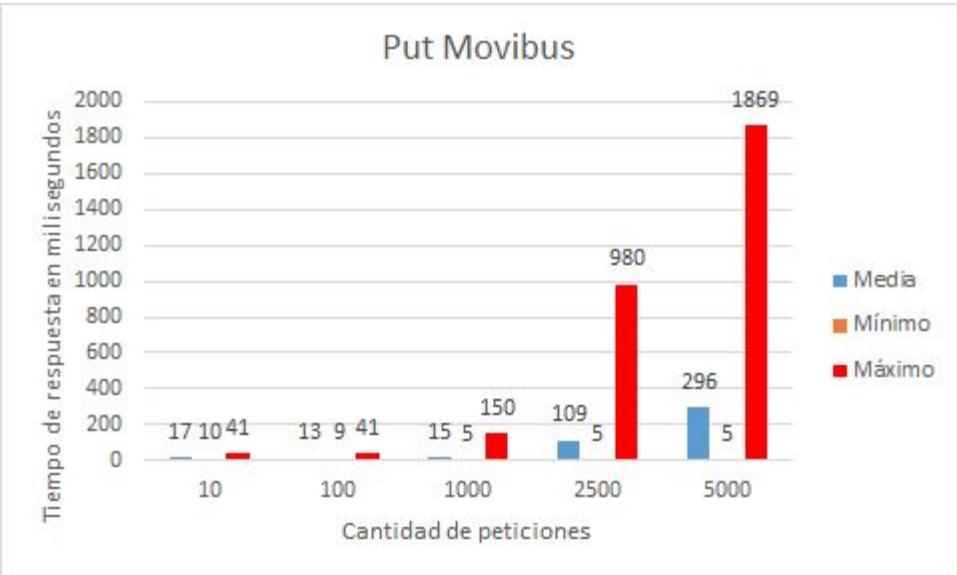
Get EstaciónVcub

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	-							



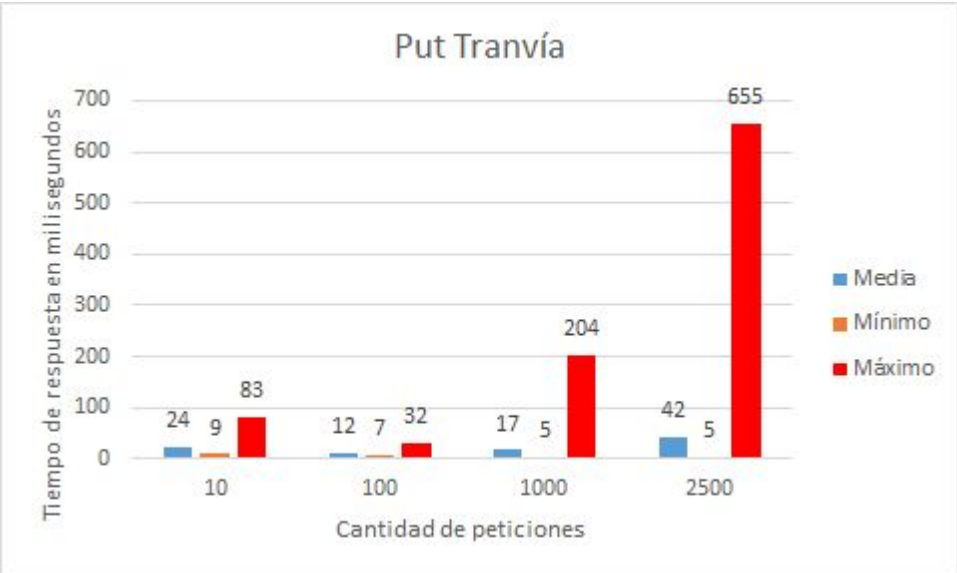
Put Movibus

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



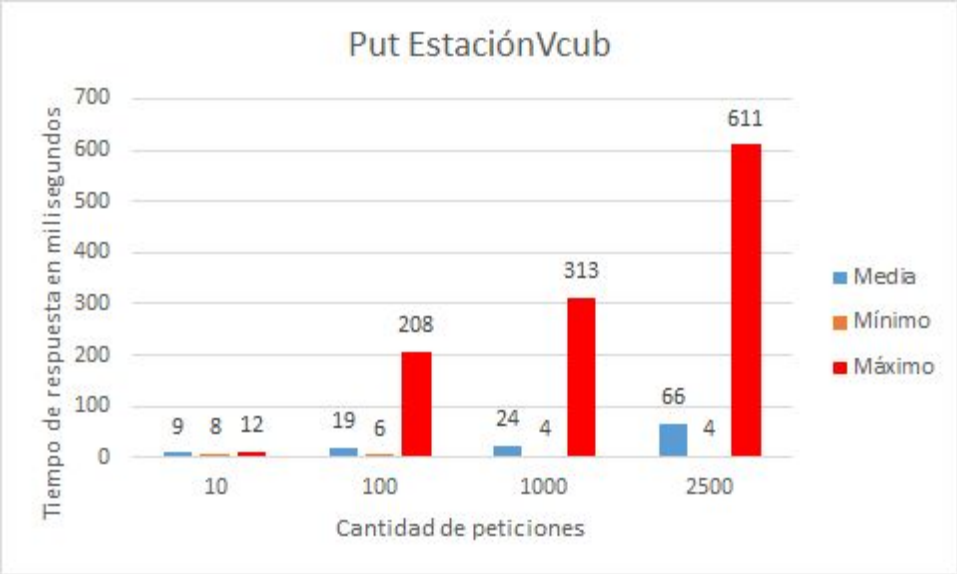
Put Tranvía

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	-							



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.