



Experimento Numero 1 Final

12.03.2017

Juan Pablo Otalora cod: 201425525

Ana María Espinosa cod: 201425031

Fabio Andrés López cod: 201423782

Grupo numero 2

Universidad de los Andes

Bogotá, Colombia

Vision

Este documento se basa en la idea de comparar y ver los cambios significativos que obtuvimos del experimento 1 con el segundo experimento. Se debe analizar las modificaciones al proyecto con las nuevas adiciones que estas conllevan. Se comparan resultados de pruebas y la diferencia en la tecnología y sus respuestas.

Objetivos

1. Observar los nuevos cambios al experimento 2 y que conlleva en estos.
2. Comparar estos resultados con los del primer experimento para saber qué cambia en las pruebas al agregar más requerimientos.

Especificaciones


Para el inicio de este documento cabe la pena resaltar los cambios que deben ser acentuados para el desarrollo de esta segunda parte. Primero que nada debe ser claro que en este proyecto se le agregaba un componente eléctrico, que será quien mande las medidas y este debe ser usado usando el componente de Node Red. Además se nos pide agregarle el componente de disponibilidad a todo el experimento, ya que para el caso de estudio este es muy necesario, hay vidas en juego.

Conclusión acerca de Especificaciones

Sin duda alguna estos nuevos requerimientos conlleva a un mayor número de cambios que deben ser vigilados muy cercanamente para no afectar la funcionalidad de la aplicación. Debemos continuar con los atributos de calidad pedidos anteriormente.

Problematica:

Para comenzar se debe contextualizar el caso de desarrollo, en este caso tenemos un sistema de emergencias y de registros de brazaletes cardiacos los cuales son absolutamente necesarios para la supervivencia de nuestros pacientes. Es por esto que en este nuevo experimento debemos asegurar la mayor disponibilidad de



nuestro proyecto, si se llegara a caer la pagina o si ésta dejara de funcionar muchas vidas estarían en riesgo y no queremos que esto pase. Desde el otro lado los microcontroladores deben ser controlados de manera específica y deben funcionar perfectamente para asegurarse de no tener lecturas erróneas ya que estas pueden afectar muy negativamente el diagnóstico real.

Hipotesis

En este caso se nos pide que el web app continúe respetando todos los atributos de calidad anteriores, tales como la priorización de eventos y el desempeño y además le agregamos el componente de disponibilidad.

Descripción del experimento

En el experimento queremos concluir si podemos maximizar la vida útil de nuestro web app y si además le podemos agregar el componente de microcontroladores que se encargan de mandar las señales pertinentes al hospital.

Requisitos para el desarrollo del experimento

I. Artefactos a construir

Para la construcción de la aplicación se usa la estructura básica requerida para el uso del framework Play. Por parte del framework se estarán usando los Streams de Akka para manejar transacciones asíncronas, junto a librerías en Scala que posteriormente manejarán diferentes elementos de la base de datos y de los templates. Así mismo es clave el uso de SBT, herramienta que maneja las dependencias y permitirá generar el build de la aplicación.

Es necesario desarrollar, bajo la misma estructura estándar, diferentes módulos haciendo uso de las APIs que se están implementando. Para contestar los servicios HTTP está el archivo de routes (ubicado en conf/routes dentro de la carpeta madre del proyecto), quien se encarga de asociar el request generado por el cliente con un método desarrollado en el controller del recurso. Estos controllers serían uno por cada módulo a usar dentro de la aplicación (inicialmente todas las entidades). El controller se comunica con la parte de lógica, donde se almacenan los archivos .java que representan a las diferentes entidades lógicas involucradas en el modelo.

En esta etapa inicial se involucran todos estos módulos para generar los CRUD y los requerimientos necesarios.

II. Recursos de la experimentación

Para la ejecución de la aplicación se utiliza el IDE IntelliJ 2016.3.4, que despliega los servicios web de la aplicación. Previo a la experimentación se implementan los servicios HTTP requeridos desde el complemento de google chrome POSTMAN para probar el funcionamiento correcto de la aplicación. El despliegue se realiza en un equipo con un procesador Intel de 2.60GHz, con una memoria RAM de 8GB en un sistema operativo de 64 bits. Estas pruebas hacen uso de la herramienta Apache JMeter (3.1.r1770033).

Planeación y resultados esperados

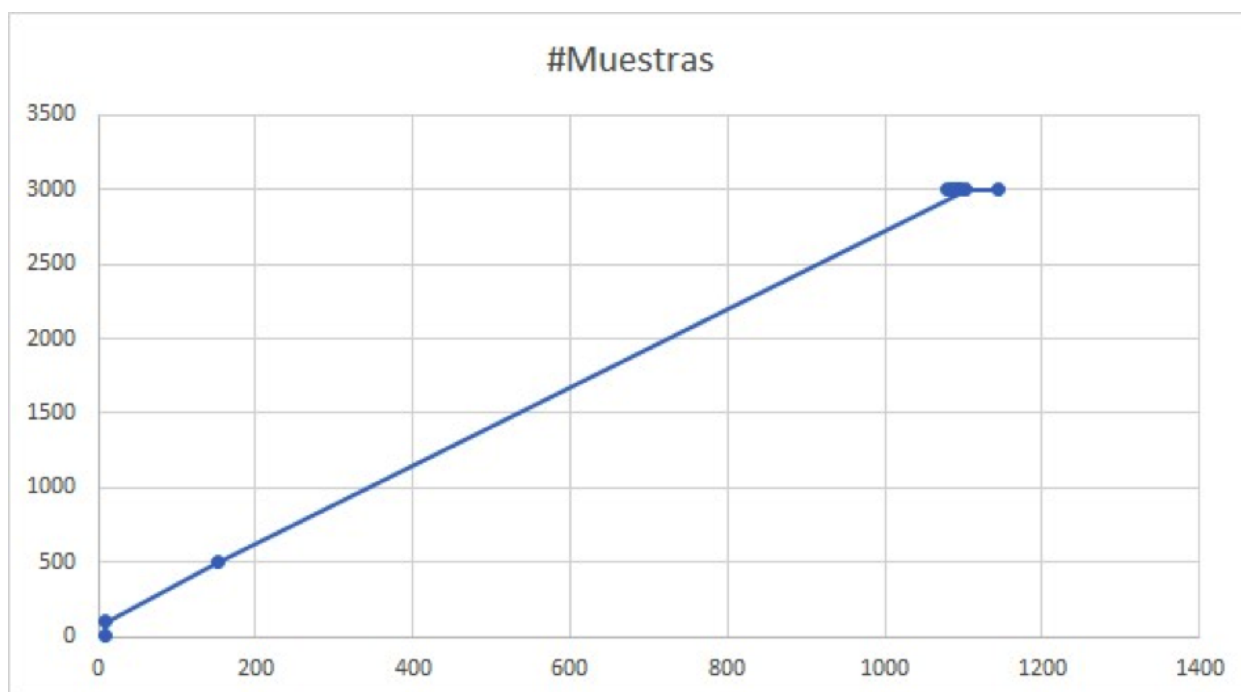
Para una configuración de 3000 peticiones con un ramp-up de un segundo se espera un comportamiento que no exceda los dos segundos. En cada etapa del experimento se irá aumentando el número de peticiones gradualmente hasta llegar al número esperado. Todos los datos se anotarán y se verá el comportamiento de la aplicación ante el número máximo de peticiones esperado. Además a esto estamos buscando que la web app no se caiga y que podamos asegurar que se mantenga disponible, unido a esto queremos que los microcontroladores manden las señales pertinentes y que estas se puedan procesar de manera correcta.

Resultados obtenidos

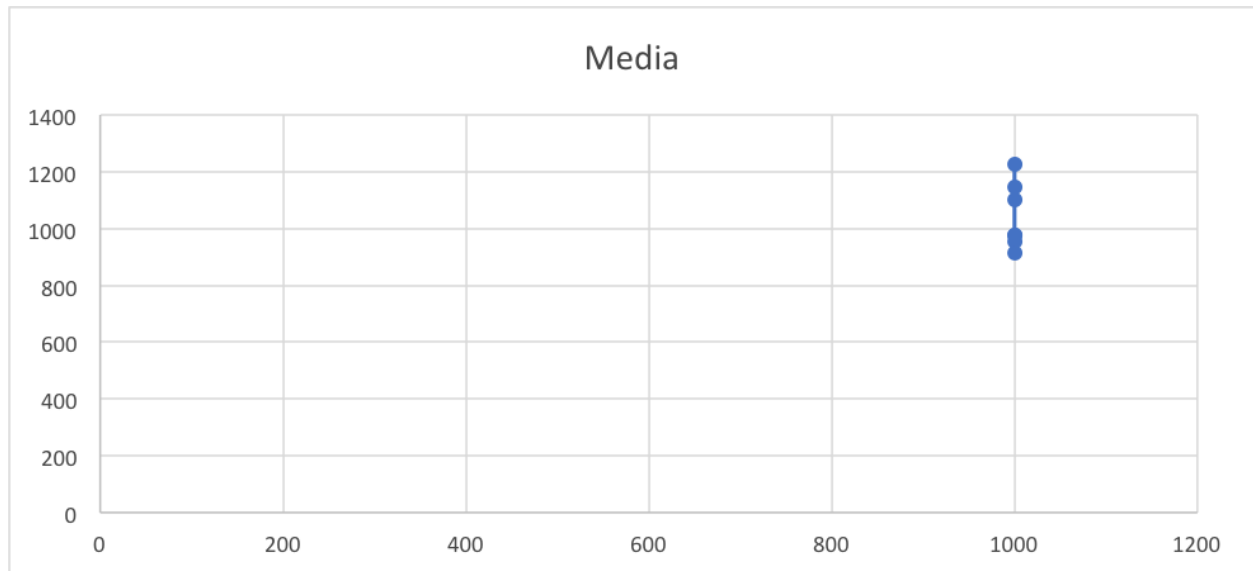
Debido a la organización de nuestro web app y unido a un desarrollo en el cual se trata de simplificar errores y mantenerlos a un mínimo aceptable nos damos cuenta que la aplicación si puede mantener un porcentaje de disponibilidad mayor al 99% lo cual asegura que la aplicación se mantenga una cantidad de tiempo aceptable por nuestro mercado.

Visto desde las nuevas pruebas que se hicieron sobre el experimento 1 Final encontramos esto:

Aqui tenemos los resultados del primer experimento parcial:



En esta primera gráfica estamos comprando los tiempos de ramp up que teníamos en la primera parte de esta experimentación. Aquí vemos el número de threads en comparación con el tiempo de ejecución en milisegundos. Para valores bajos del número de muestras (de 500 o menos) se tiene un tiempo inferior al de 200 milisegundos, un resultado ideal. Sin embargo, se nota un crecimiento lineal al aumentar el número de threads. Al llegar al número deseado (3000) se realizan varias pruebas, las cuales muestran un comportamiento constante del tiempo de respuesta, que siempre ronda los 1100 milisegundos.



En este segundo caso observamos comienza ya en un segundo las pruebas, pero en este caso se mantiene altamente escalable ya que no tiene una pendiente. Aquí nos damos cuenta cómo ha mejorado la escalabilidad de nuestra aplicación.

En forma de comparación sabemos que en el primer experimento se comenzaba con un mejor tiempo pero no escalaba tan fácil como en el segundo. Más aún en el primero concluimos que se debía mejorar la escalabilidad y es esto fue cierto para el término de nuestro experimento.

Conclusiones

- Los elementos arquitectónicos de actores permiten un funcionamiento más eficiente de los proyectos y esto afecta la disponibilidad de una aplicación que debe ser usada a toda hora.
- Debemos mejorar la disponibilidad igualmente llegar a un número parecido a un cuatro sigma ayudaría enormemente con el número de down-time de la aplicación.