EXPERIMENTO 1

Actualmente, estamos en un proyecto de Desarrollo de Software para la Fundación Santafé de Bogotá apoyado por el Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes. Para la correcta realización del proyecto, se definieron motivadores de negocio y atributos de calidad que deben ser cumplidos al culminar el desarrollo del proyecto de Software. El grupo Código Andino, desea realizar el proyecto planteando una correcta arquitectura de Software y guiados por patrones de desarrollo orientados al cumplimiento de los escenarios de calidad especificados.

Se hace uso de Experimentos, para validar las decisiones arquitecturales, su cumplimiento con escenarios de calidad y hacer una detección temprana de errores en el diseño del proyecto. También permite comprender el impacto que tendrá la arquitectura evaluada, verificar el funcionamiento de librerías externas y esbozar el funcionamiento ideal de la aplicación.

El análisis del experimento se hará básicamente en tres fases fundamentales: pre-experimentación, experimentación y post-experimentación. En la primera se describirá la planeación del experimento, en la segunda se documentará el proceso de experimentación y en la última la medición y estudios de los resultados obtenidos para llegar a conclusiones del diseño.

PRE-EXPERIMENTACIÓN

PROBLEMÁTICA

La fundación Santa Fe quiere hacer un seguimiento a sus pacientes con problemas de tensión y peso; desea poder monitorearlos de cerca y actuar con mayor rapidez dado un caso de emergencia. Como hipótesis, componentes de Software construidos en lenguaje JAVA se pueden comunicar de forma remota en un tiempo limitado y con concurrencia de casos.

OBJETIVO DEL EXPERIMENTO

La arquitectura busca conectar una aplicación web (Servidor) con usuarios (Clientes) de forma remota (simulado por una aplicación Java). Se espera que los usuarios puedan enviar sus datos de tensión y peso. En caso de datos importantes para el negocio responder al estimulo de los datos.

DESCRIPCIÓN

En la aplicación de los usuarios se conecta al servidor y le envía los datos necesarios para cada consulta que realice; si es IMC envía el peso actual y si quiere garantizar su altura actual también envía su altura; de lo contrario es usada la altura que tiene el usuario en el registro; con esto el servidor responde con una serie de consejos, si estos son requeridos; o con algo que indique que todo esta en orden. En cuanto al envío de la tensión, se envía al servidor 3 valores para que aclare si esta o no dentro de una rango normal. Dependiendo de eso el servidor responde con una emergencia o un aviso normal.

Como actividades del experimento: Plantear diseño de arquitectura del proyecto, Preparar artefactos para cumplir objetivo, hacer uso de herramientas de medición y obtención de datos, Análisis de los datos y Conclusiones de lo realizado.

Como datos a Recoger: Se espera tomar tiempos de latencia y de respuesta de los artefactos elaborados.

ARTEFACTOS A CONSTRUIR

Se hará uso de un servidor que manejará de manera concurrente las peticiones de los usuarios, usando JEE. Además se hará una aplicación remota que simule el artefacto usado por el cliente para enviar sus datos. Una aplicación web que será la que despliegue los resultados y desde la cual los médicos podrán observar a sus pacientes (Dashboards).

RECURSOS DE LA EXPERIMENTACIÓN

Se trabajará sobre NetBeans y con un servidor GlassFish; la aplicación de los usuarios se modelará en Swing y se conectará con el servidor. Los Dashboards serán desplegados como una aplicación web, haciendo uso de XHTML y ICEFACES.

RESULTADOS ESPERADOS

Se manejará una arquitectura SOA que maneja las peticiones con REST. Se espera que el Servidor GlassFish logre hacer el manejo de peticiones con más de 600 usuarios. Sin que sobrepase 1,5 segundos de petición. Además, se espera que sea posible interconectar componentes en distintos nodos y hacer comunicación entre ellos.

DURACIÓN Y ETAPAS

Propuesta a Evaluar y Descripción del documento del experimento 1. Tiempo Estimado: 1 Semana

Implementar la aplicación Swing. Tiempo Estimado: 1 Semana

Implementar la parte gráfica de la aplicación JEE. Tiempo Estimado: 2 Semanas

Implementar la aplicación JEE. Tiempo Estimado: 1 Semana

EXPERIMENTACIÓN

Para hacer la experimentación, tuvimos que construir los artefactos ya mencionados. Los Requerimientos Funcionales propuestos a probar (Avance del Paciente, Recordatorios y Manejo de Emergencia) los desarrollamos de la siguiente forma:

El avance del paciente es visualizado por la parte gráfica del Servidor, actualmente son datos simulados de médicos y pacientes. Los datos son manejados por un Bean que representa la clase persistente del Hospital con todos los datos necesarios y permite la creación, actualización, eliminación y búsqueda de datos.

Para realizar Recordatorios, tomamos una decisión de implementación, la aplicación –es decir, el usuario móvil- tendrá que recordar las actualizaciones y desplegarlas en el momento justo en el mismo.

Es decir, el servidor no se preocupa por enviar recordatorios, sino que es la aplicación remota la que recuerda al usuario la toma de datos. Así que, este módulo se desarrollo en la aplicación StandAlone.

Finalmente, para el manejo de emergencias, que incluye registro de datos, se desarrollo un paquete en el Servidor encargado de escuchar las peticiones (que pueden ser generadas por la aplicación StandAlone) y reorientarlas a este paquete, este toma la información que viene en un REST, y por tanto es de un paciente, y hace los llamados respectivos al Bean de autentificación, este a su vez hace llamado a los servicios de la aplicación para registrar la información y espera una respuesta por parte de la lógica de negocio. Esta respuesta, puede ser de tipo "OK" o "ALERT", y en caso, que se produzca la última, se retorna al llamado de la petición las indicaciones, según la lógica de negocio.

Para probar los requerimientos no funcionales

POST-EXPERIMENTACIÓN

RESULTADOS OBTENIDOS

LATENCIA

Para obtener resultados en Latencia, hicimos mediciones empíricas, es decir, realizamos mediciones físicas de tiempo, validando que no sobrepasará el tiempo de 3 segundos y 1.5 segundos según sea el caso.

Procesamiento de la información

Hicimos 20 medidas desde el lanzamiento de los datos a agregar hasta la correcta visualización en el DashBoard, el dato máximo, que representa el peor caso, se registro en 1,239 segundos.

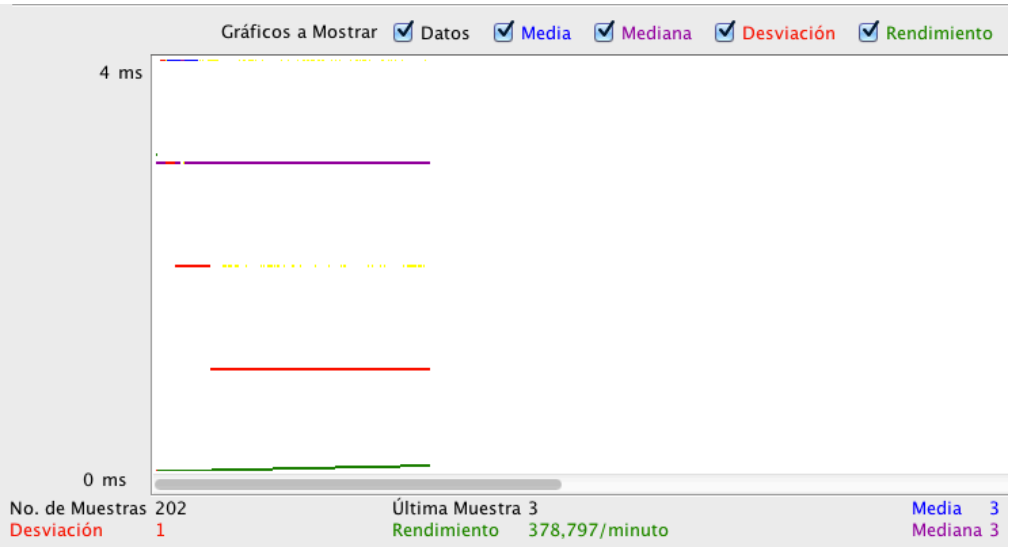
Atención a Emergencias

Para esta toma de datos, consideramos el tiempo desde que se envío los datos de la aplicación remota hasta que regreso, pasando por los requerimientos lógicos de la información. Nuevamente, el tiempo máximo, obtenido para 20 mediciones de distintos tipos de datos, fue de 1,032 segundos.

ESCABILIDAD

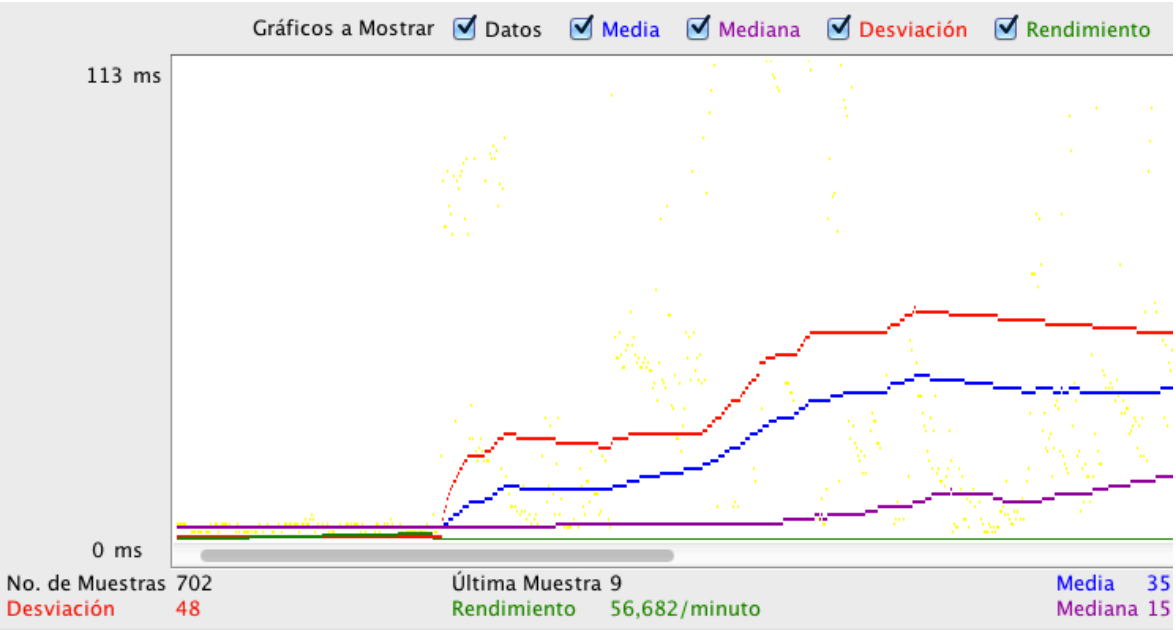
Hicimos uso de JMeter, para generar resultados de desempeño y escalabilidad válidos. Estos fueron los datos:

Para 200 usuarios, con peticiones concurrentes, separadas por un RampUp de 3.



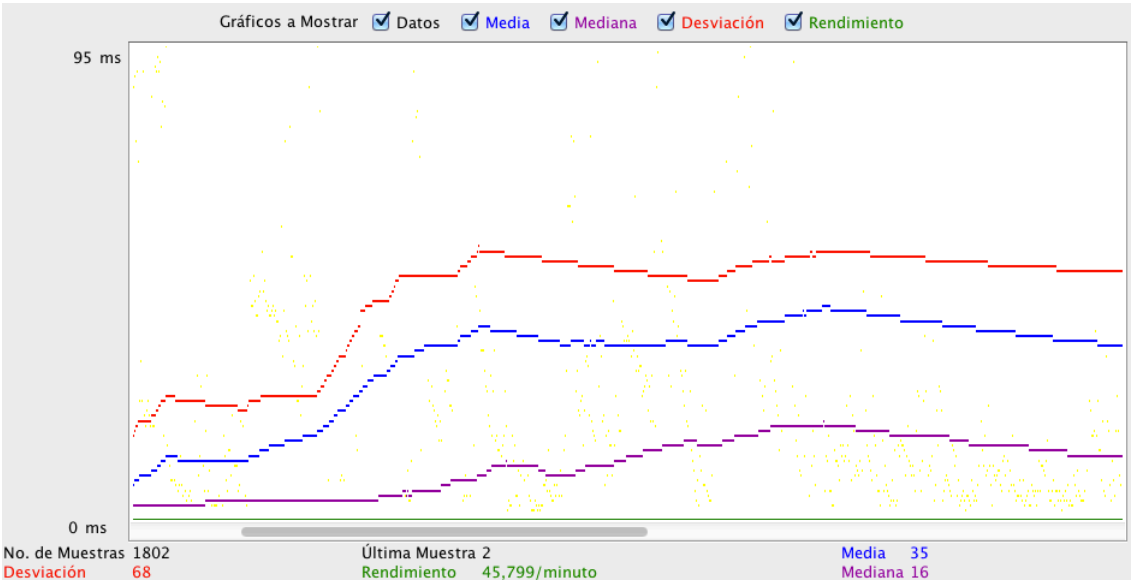
Gráfica 1

Para 700 usuarios y un Rampup de 2.



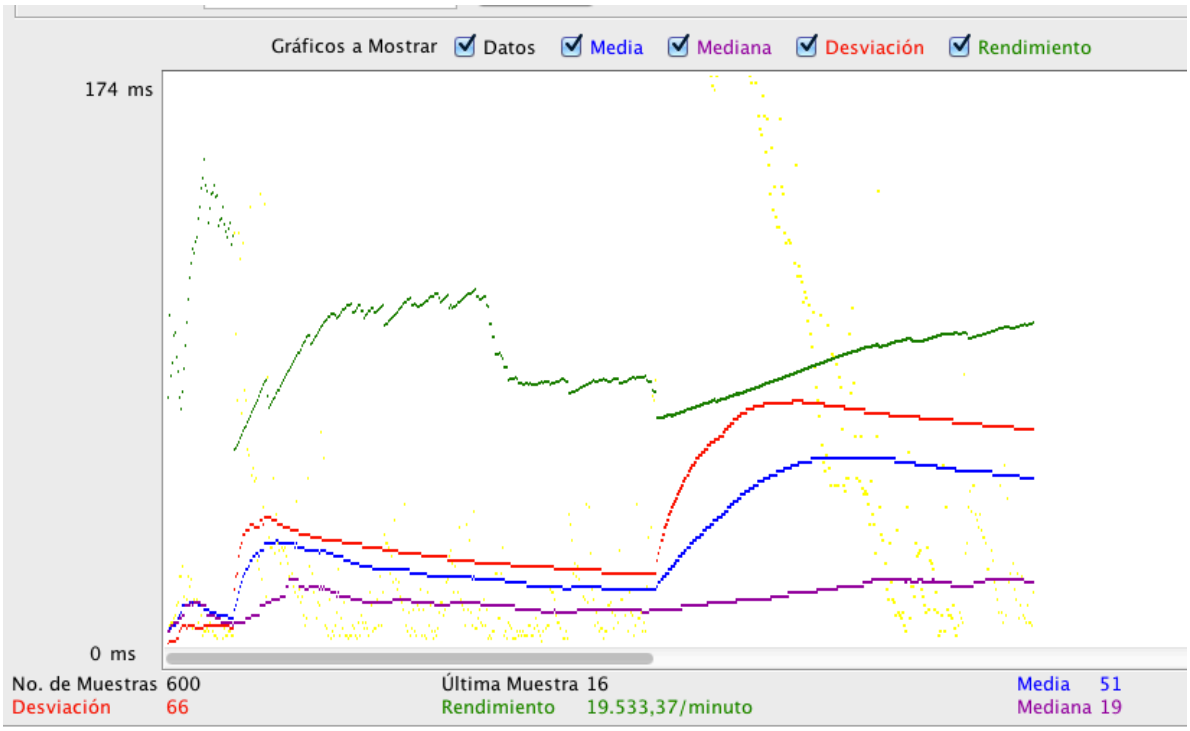
Gráfica 2

Luego, número de usuarios 1800 y un intervalo de separación de petición de 1.



Gráfica 3

Por último, número de usuarios 600 y Rampup de 1.



Gráfica 4

DURACIÓN REAL

Dado que no se puso en marcha el experimento 1 desde la fecha de Inicio estipulada, realizamos el experimento de la siguiente forma: Propuesta a Evaluar en una Semana, Implementar aplicación Swing, implementar Dashboards y JEE en una Semana. En esa semana finalizo el tiempo del Experimento, así que los análisis sobre el trabajo se hicieron en la última semana. Duración Real, dos semanas.

ARTEFACTOS CONSTRUIDOS

Todos los artefactos fueron construidos para el experimento, asegurando el material para poder realizar las medidas. Al final fueron hechos: una aplicación Standalone –simulador de Paciente- y un proyecto en NetBeans que maneja el Servidor y la parte HTML de la aplicación.

ANÁLISIS

LATENCIA

Fue considerado más importante que el promedio de los datos, el dato que representará el peor caso de ejecución, puesto que, se debe asegurar PARA TODA petición un registro en menos de 3 segundos. Si existe una que no lo cumple, no se lograría asegurar este requerimiento. Para las pruebas que realizamos, el máximo dato fue de 1,239 segundos. Claramente, es un resultado menor a 3 segundos. Luego, la implementación de un Servidor Principal que fuera accedido en pocos pasos por la petición provoco que este resultado cumpliera aquel requerimiento no funcional.

Del mismo modo, el análisis para casos de Emergencia, el dato máximo fue de 1,032 segundos. Por supuesto, es mucho más bajo que 1,5 segundos. Inclusive, un dato atípico en la muestra de datos tomada. Este resultado, puede ser atribuido al método de envío de datos, la información compacta y liviana, los pocos llamados de métodos para el registro de datos, y el manejo de estructuras de datos como HashMap y ArrayList.

ESCALABILIDAD

Muy satisfactorios fueron los datos que obtuvimos, puesto que, ninguno supero 0,002 segundos de registro de datos. En la gráfica 1, para 200 usuarios y con un intervalo de tiempo de petición igual a 0,015 segundos de separación, los resultados fueron lineales y no hay evidencia de efecto tamaño en el tiempo de respuesta. Muy diferente a la gráfica 4, con 600 usuarios y un intervalo de tiempo de 0,0016666667, que mostro un cambio abrupto desde 350 usuarios pero se fue normalizando. Explicamos este fenómeno por la reestructuración de la estructura de datos HashMap, para manejar todos los usuarios nuevos. Por último, hicimos una exageración en el número de usuarios y el tiempo de distancia entre peticiones con datos de 1800 usuarios y 0,00055555556 segundos. Ninguna petición supero 90 ms, 0,09 segundos. Esto lo podemos atribuir a la aplicación Java EE, y a las prácticas que emplea para manejar este tipo de concurrencia.

CONCLUSIONES

Finalmente podemos concluir que el experimento fue de mucho provecho y las hipótesis fueron acertadas en su gran medida. Aunque el experimento se haya empezado con algún tiempo de trazo, se

concluyo exitosamente y se realizando las pruebas necesarias para lograr obtener resultados necesarios la posterior implementación de aplicaciones apoyadas sobre patrones de diseño.

Inicialmente fue difícil entender cual arquitectura era correcta para nuestra experimento, pero al entender que para la implementación de protocolo de comunicación utilizado era necesario manejar una arquitectura SOA para lograr representar correctamente y de manera completa la complejidad de nuestro diseño, se logro ver de manera global toda la estructura arquitectural necesaria para ser implementado el experimento.

En cuanto a los resultados de latencia y escalabilidad, fueron muy satisfactorios, la respuesta del servidor a la aplicación eran casi inmediatos, y no se podían distinguir que corrieran en maquinas diferentes. Lastimosamente el dashboard no actualiza correctamente y aunque la medida por humanos no es la mas certera, se da tranquilidad de que la latencia no es previsible por el usuario. En cuanto a las pruebas de escalabilidad, fue satisfactorio ver que las peticiones aun con un ramp-up muy bajo no se demoraban mas de 1 segundo en ser respondidas.