**DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**

**ARQUITECTURA Y DISEÑO DE SOFTWARE (ISIS2503)**

**PROFESORA: PhD KELLY GARCÉS**

**PRESENTADO POR:**

**ANDRÉS CAMILO ZULETA – 201413405**

**CAMILO CARRILLO – 201215749**

**SERGIO MIGUEL MADERA – 201417957**

**FECHA DE PRESENTACIÓN:**

**17 DE FEBRERO DE 2016**

**UNIVERSIDAD DE LOS ANDES**

**RESUMEN REJECUTIVO:**

Dentro del presente documento se encuentra el trabajo realizado por el grupo de trabajo número cinco para la entrega preliminar del primer experimento correspondiente al curso de Arquitectura y diseño de software. Este documento incluye los siguientes anexos. En primer lugar se encuentra el tipo de arquitectura seleccionada para realizar el experimento y la justificación de dicha elección en aras de los requerimientos no funcionales planteados para este experimento (desempeño y escalabilidad).Y en segundo lugar encontraremos el planeamiento, desarrollo y ejecución de los experimentos y pruebas de los atributos de calidad realizados sobre el sistema SATT construido.

**ARQUITECTURA SELECCIONADA Y JUSTIFICACIÓN:**

La arquitectura seleccionada para la realización del proyecto, al menos de manera provisional,

Es la arquitectura basada en servicios REST de JAX-RS. Dicha decisión fue tomada inicialmente debido a la facilidad de manejo que esta otorga y por el potencial que esta le otorga al equipo desarrollador para desarrollar, valga la redundancia, de manera eficaz y ágil. Adicionalmente esta arquitectura cuenta con los beneficios que REST tiene para ofrecer y su separación por capaz permite un fácil manejo de sus distintos componentes, facilitando así su mantenimiento en el caso de que se deseara una longeva esperanza de vida para la aplicación y además su fácil modificación en caso de ser requerida.

Ahora, esta arquitectura resulta también beneficiosa para los objetivos del experimento, es decir, el cumplimiento de los atributos de calidad de desempeño y escalabilidad. Nos podemos dar cuenta de esto gracias a los resultados obtenidos durante el laboratorio de desempeño y escalabilidad llevado a cabo durante el curso, donde se pudo ver que una aplicación REST soportaba de manera eficaz tanto el desempeño como la escalabilidad para un número significante de usuarios (más de diez mil). Por tanto a partir de esto se puede deducir que JAX-RS permite manejar de manera adecuada el número de solicitudes que se espera para la aplicación realizada durante el experimento (alrededor de cuatro mil).

Sin embargo también es menester mencionar que el grupo de trabajo planea desarrollar en paralelo la misma aplicación utilizando una arquitectura de tipo play! Y comparar los resultados entre ambas arquitecturas para que al momento de la entrega final se escoja aquella arquitectura con la cual se obtengan los resultados más satisfactorios.

**DISEÑO Y DESARROLLO DE LAS PRUEBAS.**

**Pre-experimentación**

1. Problemática: Se espera probar que el sistema, en su estado actual, puede manejar dos tipos de operaciones distintas, cumpliendo con ciertos escenarios de calidad asociados al desempeño. En este contexto hablamos de un sistema que recibe notificaciones de estado y de eventos sísmicos, para poder generar boletines de alerta, lo cual implica un alto nivel de urgencia y por eso se plantean las siguientes latencias. En el caso de la recepción de mensajes y notificaciones de eventos sísmicos se espera alcanzar una latencia menor a quinientos milisegundos (para cuatro mil hilos de ejecución de manera concurrente), al igual que para la generación de boletines de alerta. Nuestra hipótesis es que se pueden alcanzar las latencias anteriormente propuestas para los distintos escenarios de calidad.
2. Objetivo del experimento: Se pondrá a prueba la hipótesis por medio de simulaciones de carga y estrés, usando la aplicación Apache JMeter. Esta no sólo permite modelar escenarios específicos con tiempos y número de hilos de ejecución dados, sino que también permite deducir datos resultantes en el experimento, como por ejemplo el tiempo de latencia máximo de las transacciones.
3. Descripción del experimento: Se utilizará JMeter para modelar los dos contextos planteados en los dos escenarios de calidad. Con la misma herramienta se ejecutarán los modelos y se obtendrán los datos resultantes. De estos datos, para los dos escenarios, tomaremos la latencia por transacción máxima, que nos permitirá deducir el tiempo máximo que tomó llevar a cabo una transacción, y lo compararemos con el tiempo deseado para afirmar si se cumple el escenario de calidad o no. Adicionalmente para poder deducir el tiempo total de ejecución de toda la prueba lo que se hará es coger el tiempo de ejecución promedio de los hilos y multiplicarlo por la cantidad de hilos que se usaron para realizar la prueba.
4. Artefactos a construir: A parte del software desarrollado (SATT) y las pruebas en Apache JMeter no se dispuso de algún elemento adicional.
5. Recursos de la experimentación: Para el despliegue de la aplicación se usará NetBeans IDE 8.1 y GlassFish Server 4.1.1. Para la ejecución de la prueba se usará Apache JMeter 2.13.
6. Resultados esperados: Se espera que las latencias máximas para cada escenario sean menores a las establecidas, para todos los casos la latencia máxima es de quinientos milisegundos.
7. Duración y etapas: Se entregarán una serie de pantallazos mostrando los resultados de cada una de las pruebas con sus escenarios planteados. La ejecución de cada una de las pruebas no ha de tomar más de cinco minutos. Se llevará a cabo cada una de las pruebas por separado.

**Post-experimentación**

En esta fase se realiza el análisis del experimento. Es imperativo tener claro lo siguiente:

1. Resultados obtenidos

C:\Users\ASUS\AppData\Local\Microsoft\Windows\INetCache\Content.Word\4000tpara100sensores.png

Prueba 1: Latencia máxima de 500 milisegundos, 4000 solicitudes de enviar mensaje con latitud, longitud, velocidad y altura de los sensores. El resultado fue 80 milisegundos como latencia máxima (0,08 segundos), lo que quiere decir que el escenario de calidad se cumple, pues la latencia máxima fue menor a 500 milisegundos. Cabe resaltar que este servicio se comporta diferente a diferente número de threads, para información más detallada por favor consultar la carpeta de imágenes anexa al proyecto o las tablas y gráficas de Excel también anexas al proyecto. Para esta prueba es necesario mencionar que el comportamiento de la misma también es relativo a la cantidad de sensores que se posean, como anteriormente se dijo esto puede observase con mayor detalle en la sección de documentos anexos al proyecto.

C:\Users\ASUS\AppData\Local\Microsoft\Windows\INetCache\Content.Word\para4000.png

Prueba 2: Latencia en la generación de boletines menor a 500 milisegundos. El resultado que obtuvo para la generación de 1 boletín fue de un máximo de 154 milisegundos y el boletín se retorna como un objeto JSON. También es necesario decir que después de cierto número de threads el sservicio comienza a presentar errores aunque en una proporción demasiado baja (inferior al 1%) por lo cual estos errores no fueron tenidos en cuenta.

Vale la pena aclarar que el estímulo de la prueba corresponde a un thread, ya que el boletín de alerta se genera una vez se registra un evento sísmico, por lo cual no es acorde con la realidad realizar una prueba como la anterior, donde se generen 4000 eventos sísmicos para producir 4000 boletines.

1. Duración real: Para la prueba No.1 el tiempo de ejecución total fue de 3 minutos 14 segundos. Esto fue inferior al tiempo planteado de 5 minutos, el cual se estimó aleatoriamente con base en ejecuciones previas. De la misma manera, la prueba del segundo escenario no tomó más de 1 minuto, por lo cual se puede afirmar que se tuvo un tiempo óptimo de realización de pruebas.
2. Artefactos construidos: Los únicos artefactos construidos fueron los escenarios en JMeter. No se llevó a cabo ninguna otra construcción pues las herramientas disponibles fueron suficientes para modelar los escenarios con todas sus cualidades establecidas y poder llevar a cabo las pruebas necesarias.
3. Análisis: Podemos argumentar que la arquitectura escogida fue favorecedora para los escenarios de calidad que se deseaba alcanzar. Podemos afirmar esto pues estos escenarios requerían valores menores a ciertos máximos de latencia, para contextos específicos, y se pudo mantener los tiempos de respuesta por debajo de los máximos establecidos. Esto se debe a que la arquitectura JAX-RS, por su estructura, permite el acceso directo a los recursos necesarios para llevar a cabo la transacción, que aunque puede ser costoso a nivel de consumo de recursos favoreció el desempeño de la aplicación. En caso de manipularse información o modelos más pesados sería necesario considerar una arquitectura diferente, pero en este caso se manipulan objetos simples tales como los mensajes y boletines, que con pocos datos permiten el manejo de la lógica de todo el sistema.
4. Conclusiones: En conclusión el experimento fue exitoso y se puede afirmar que, para los escenarios de calidad establecidos, la aplicación tiene un buen desempeño. La arquitectura escogida y el diseño implementado permiten obtener una aplicación capaz de cumplir con los requerimientos de calidad estipulados en este experimento y se espera que tenga un buen desempeño en futuros eventos. Para esto se pensó en la implementación de EJB’s sencillos, modelos desacoplados y servicios atómicos. Todo esto es una estrategia que nos permitirá modificar el proyecto acorde a nuevos requerimientos con un mínimo impacto en los ya cumplidos.