UNIVERSIDAD DE LOS ANDES FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

SISTEMA DE ALERTA TEMPRANA DE TSUNAMIS (SATT)

REPORTE ENTREGA PARCIAL EXPERIMENTO 2

Equipo de Trabajo:

Grupo 5

Integrantes:

Andrés Camilo Zuleta Calderón ac.zuleta10@uniandes.edu.co

Sergio Miguel Madera sm.madera10[@uniandes.edu.co](mailto:and-gome@uniandes.edu.co)

Camilo Carrillo ca.carrillo2369@uniandes.edu.co

Tabla de contenido

[RESUMEN EJECUTIVO 2](#_Toc448434597)

[ARQUITECTURA SELECCIONADA Y JUSTIFICACIÓN 2](#_Toc448434598)

[**Decisiones de arquitectura** 2](#_Toc448434599)

[Desempeño 2](#_Toc448434600)

[Escalabilidad 3](#_Toc448434601)

[Disponibilidad 3](#_Toc448434602)

[DISEÑO Y DESARROLLO DE LAS PRUEBAS 4](#_Toc448434603)

[Pre-experimentación 4](#_Toc448434604)

[Post-experimentación 5](#_Toc448434605)

[Prueba 1 – Generación de boletín de alerta: 5](#_Toc448434606)

[Prueba 2 – Recepción información y respuesta a cambios de nivel de altura 6](#_Toc448434607)

[Prueba 3 – Escalabilidad 7](#_Toc448434608)

[Comparación resultados obtenidos Experimento 2 vs Experimento 1 8](#_Toc448434609)

[Prueba 1 – Generación de boletín de alerta: 8](#_Toc448434610)

[Prueba 2 – Recepción información y respuesta a cambios de nivel de altura 8](#_Toc448434611)

[Prueba 3 – Escalabilidad: 8](#_Toc448434612)

[Conclusiones 8](#_Toc448434613)

# RESUMEN EJECUTIVO

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 un análisis de las decisiones tomadas en cuanto a la arquitectura y la justificación de estas, tomando en cuenta los requerimientos no funcionales planteados para esta entrega del experimento. En adición, se describe la planeación, el desarrollo y la ejecución de los servicios 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 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. Esto podemos deducirlo a partir de los resultados que obtuvimos 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).

# 

# **Decisiones de arquitectura**

## Desempeño

Las decisiones tomadas con el grupo para satisfacer los escenarios de calidad propuestos por el enunciado que conciernen al atributo de calidad de desempeño son las siguientes.

En primer lugar se propone una arquitectura a tres capas, siendo estas la capa de presentación, la capa de servicios y finalmente la capa de lógica y persistencia. Lo anterior es un factor que contribuye significativamente al desempeño de la aplicación ya que al estar desacoplada la presentación de la aplicación de los servicios y de la lógica esto permite que la lógica y componentes que se encuentran ubicados en la capa de presentación no agreguen latencia ni afecten las solicitudes que son generadas por los usuarios.

En segundo lugar como manejador de base de datos de nuestra unidad de persistencia se utiliza MongoDB que al ser una base de datos no relacional es capaz de procesar de forma más rápida las operaciones del CRUD de los diferentes objetos y colecciones de objetos que son necesarios para el correcto funcionamiento de la lógica de la aplicación (y en general presenta un desempeño notoriamente superior en cuanto a operaciones de lectura se trata) lo cual nos apoya a satisfacer de forma efectiva los escenarios de calidad asociados al desempeño.

## Escalabilidad

Las decisiones tomadas con el grupo para satisfacer los escenarios de calidad propuestos por el enunciado que conciernen al atributo de calidad de escalabilidad son las siguientes.

En primer lugar y como anteriormente se mencionó, el grupo propone una solución utilizando una arquitectura a tres capas, capas que ya anteriormente fueron especificadas. La ventaja que esto supone en aras de la escalabilidad de la aplicación es que al estar separada la capa de presentación de los servicios y la lógica es que podemos correr dichas capas como instancias diferentes de la aplicación, es decir, podemos tratar la interfaz gráfica y la lógica como si de dos aplicaciones diferentes se tratasen y por tanto esto nos permite la posibilidad de no tener que subir a una plataforma cloud (como heroku ó AWS) toda la aplicación dentro de un solo paquete, lo cual nos generaría problemas de mantenibilidad del código a futuro y además problemas de la escalabilidad dado el alto acoplamiento entre las capas (debido a que para tener una instancia de una aplicación hecha en JEE es necesario empaquetar toda la aplicación dentro de una sola capa y generar un main que la ejecute). De esta forma al subir solamente la capa de lógica y la capa de servicios a una plataforma cloud podemos utilizar fácilmente las funcionalidades ofrecidas por esta, su alta disponibilidad, alto rendimiento y por supuesto, alta flexibilidad (lo cual es incluso mejor que la escalabilidad) y dejar que la plataforma administre dichos recursos por nosotros, liberando así al equipo de desarrolladores de ser los encargados de supervisar dichas tareas y solamente concentrándose en mantener la capa de presentación la cual estaría haciendo siempre peticiones de servicios REST a la instancia de la aplicación que se encuentra corriendo en la nube.

En segundo lugar en esta ocasión debemos volver a mencionar el manejador de base de datos utilizado por el grupo, MongoDb, pues este manejador ofrece dentro las posibilidades de sus configuraciones la capacidad de crear clusters de instancias de bases de datos lo cual nos permite ofrecer la capacidad de escalar con facilidad ya que los clusters de bases de datos permiten un procesamiento más rápido de los datos que son allí guardados y optimiza las operaciones de CRUD que son realizadas en los objetos de las colecciones que allí se encuentran.

## Disponibilidad

Las decisiones tomadas con el grupo para satisfacer los escenarios de calidad propuestos por el enunciado que conciernen al atributo de calidad de disponibilidad son las siguientes.

En primer lugar para asegurar la alta disponibilidad de los servidores encargados de resolver las solicitudes generadas por los usuarios de la aplicación fue el uso de un balanceador de carga que distribuyera de forma equitativa las solicitudes. Dicho balanceador de carga es un factor decisivo para poder satisfacer este atributo de calidad. Lo anterior debido a que la inclusión de dicho balanceador dentro de nuestra arquitectura nos permite tener varias instancias de nuestra aplicación corriendo en distintos servidores, esto nos permite tener una mayor confianza de que nuestro sistema será de alta disponibilidad puesto que al tener una mayor cantidad de servidores disponibles para solucionar solicitudes tendremos la confianza de que en caso de que algún servidor llegue a estar indisponible, ya sea por motivos técnicos, de sobrecarga del servidor por culpa de la alta demanda en las solicitudes y por fallos humanos, tendremos otra instancia de nuestra aplicación que se encargará de solucionar las peticiones que fueron asignadas a otro servidor. Todo lo anterior claramente soportado por el balanceador de carga el cual fue configurado por el grupo para que, en primer lugar las peticiones fueran direccionadas al servidor que poseía la mayor cantidad de recursos para resolver peticiones (directiva least\_conn de nginx), guardará en cache las sesiones de los usuarios para resolver más rápidamente sus solicitudes (sticky\_cookies) y finalmente en caso de que un servidor quedara inhabilitado, al momento de la recuperación de este no le llegaran solicitudes de forma inmediata con el fin de evitar una posible caída de dicho servidor por culpa de una saturación por alta demanda en la cantidad de solicitudes recolectadas ( directiva slow\_time). Con todo lo anteriormente dicho es la forma en la que garantizamos la alta disponibilidad de nuestro sistema a nivel de los servidores encargados de resolver las peticiones de los usuarios.

Adicionalmente la inclusión del balanceador de carga nos permite escalar de forma horizontal nuestra aplicación pues como se mencionó este distribuye de forma óptima las peticiones entre los servidores, o sea que entre más servidores tengamos obtendremos una mejora sustancial en la escalabilidad del sistema.

En segundo lugar volveremos a mencionar el uso del manejador de base de datos MongoDb quien es esta ocasión al ofrecer con facilidad la capacidad de configurar conjuntos de réplicas de las instancias de la base de datos brinda la posibilidad de tener una instancia de base de datos en cada servidor que esté resolviendo solicitudes, dichas instancias se sincronizan cada cierto tiempo (elegido por nosotros, en este caso 500ms) asegurando así la integridad y consistencia de los datos en cada una de las instancias de la aplicación y adicionalmente al ofrecer MongoDB un sistema de arbitraje en el cual existe una instancia principal que se encarga de coordinar a todas las instancias secundarias para que no se presente inconsistencia de datos y en el momento en que dicha instancia principal llegue a caer sea capaz de nombrar a una instancia secundaria como la principal de forma provisional mientras la anterior instancia principal se recupera nos garantiza que no solamente nuestro sistema tendrá una alta disponibilidad a nivel de resolución de las peticiones y de los servidores sino también a nivel del manejo y almacenamiento de los datos.

# 

# DISEÑO Y DESARROLLO DE LAS PRUEBAS

## Pre-experimentación

Se probará que los servicios que el sistema ofrece cumplen con los escenarios de calidad recibidos, incluso tras la adición de un balanceador de carga, de nuevos servidores corriendo de forma paralela diferentes instancias de la aplicación y de la creación de un conjunto de réplicas de instancias de la base de datos que tienen como finalidad preservar la consistencia y la integridad de los datos en la persistencia de la aplicación. Ha de ser posible para el sistema generar boletines de alerta con una latencia inferior a 500 milisegundos en todo momento; para probar esto, utilizaremos cuatro mil hilos con ejecución simultánea que accederán al servicio. En el mismo contexto de ejecución, el sistema ha de poder recibir información de todos los sensores con una duración total de menos de un minuto y ha de ser capaz de responder a cambios del nivel de altura con una latencia inferior a un segundo. Finalmente, debe estar dentro de las capacidades del sistema la recepción de información de 4000 sensores con el objetivo de actualizar el boletín de alerta con un tiempo de respuesta total de menos de un minuto. Esta prueba se realizará en el mismo contexto que las demás descritas anteriormente.

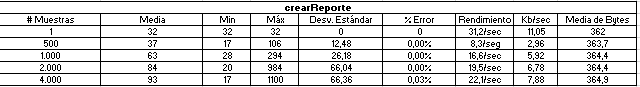
Las pruebas se realizarán utilizando JMeter. De esta forma, podemos modelar los escenarios en los que es necesario que la aplicación responda adecuadamente y analizar su desempeño con los datos reportados por JMeter. Todas las pruebas sobre los servicios consistirán de cuatro etapas, en cada una de las cuales el número de hilos que acceden de manera concurrente a cada servicio variará. En una etapa serán 500 hilos, en otra 1000, en otra 2000, y en otra 4000. Sin embargo, la prueba sobre la recepción constante de información de sensores y respuesta al cambio de altura tendrá tres etapas adicionales, diseñadas para realizar la implementación de un cluster de JMeter; en cada una hay 4000 threads recibiendo información de los sensores, y lo que variamos es el número de sensores enviando información al mismo tiempo. En cada etapa, son 100, 500 y 4000 respectivamente. Si los datos obtenidos durante cada prueba evidencian que el servicio al que corresponden cumplen con los requerimientos descritos en el párrafo anterior, será posible concluir que la aplicación como tal tiene las capacidades por el bien de las cuales fueron tomadas decisiones durante el proceso de desarrollo.

Es importante especificar además que para el despliegue de la aplicación se utilizará NetBeans IDE 8.1 y GlassFish Server 4.1.1, y que no se utilizó ningún recurso adicional además del programa desarrollado, SATT, y Apache JMeter. En la sección concerniente a la post-experimentación se mostrarán pantallazos que muestran los resultados más importantes de cada prueba, además de un breve análisis sobre los datos obtenidos y el desempeño de la aplicación durante la misma. Un análisis más detallado de lo observado sobre el experimento será anexado en un documento de Microsoft Excel al proyecto. No está de más mencionar que cada una de las pruebas, con sus etapas, se ejecutará por separado y en un ambiente aislado de los efectos que pudieran tener otras; una consecuencia importante de esto es que las tablas de la base de datos se encuentran en el mismo estado de control al inicio de cada prueba.

## Post-experimentación

A continuación, se analizarán los resultados obtenidos al realizar las pruebas.

### Prueba 1 – Generación de boletín de alerta:



La latencia máxima obtenida fue de 1100 milisegundos, y fue obtenida en la etapa con 4000 hilos concurrentes. Esta latencia es superior a la máxima tolerada de 500 milisegundos. No obstante, la latencia máxima obtenida para 1000 o menos hilos en ejecución concurrente generando el boletín de alerta fue de 151, menor a la máxima propuesta de 500 milisegundos, por lo que argüiremos a continuación que el desempeño observado es suficiente para los contextos reales a los que será sometido el sistema.

A pesar de los resultados obtenidos para la etapa de 4000 hilos concurrentes, por aquellos mencionados consideramos que el desempeño del que es capaz es suficiente en todos los casos prácticos, es decir, en todos los contextos a los que se someterá la aplicación en la realidad. Esto es porque no se generarán más de 1000 boletines de alerta simultáneamente en una situación real, dado que no se generará tal cantidad de tsunamis al mismo tiempo en la realidad. Una situación bastante posible en la realidad es, sin embargo, que deba haber más de 1000 actualizaciones sobre boletín de alerta simultáneamente, pero este servicio funciona de forma completamente satisfactoria en todas las etapas de su prueba, la cual se describirá a continuación. De esta forma, concluimos que el servicio cumple con el desempeño que necesitará durante las situaciones reales a las que será sometido.

A continuación se puede observar el análisis gráfico que fue extraído a partir de esta prueba.

Para un análisis más detallado de la prueba, por favor referirse al documento de Microsoft Excel anexo al proyecto.

### Prueba 2 – Recepción información y respuesta a cambios de nivel de altura

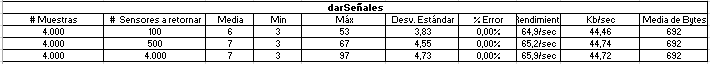


La latencia máxima al recibir la información enviada desde los sensores con frecuencia de 1 minuto y responder a los cambios del nivel de altura en la etapa con mayor carga de la prueba fue de 57 milisegundos, considerablemente inferior a la latencia máxima impuesta de 1 segundo, que equivale a 1000 milisegundos. Por esto, concluimos que la aplicación cumple con las metas impuestas al principio de la planeación, por el bien de las cuales se tomaron muchas decisiones sobre el diseño y la implementación de la aplicación. Sin embargo, no ha de subestimarse el hecho de que responder a cambios de nivel de altura puede implicar la actualización de boletín de alerta, lo que a su vez implica el acceso a la base de datos. Dado lo relativamente lento que es el mismo, esto puede convertirse en un gran obstáculo conforme la exigencia sobre la aplicación crezca y ha de ser implementado lo más eficientemente posible.

A continuación se puede observar el análisis gráfico que fue extraído a partir de esta prueba.

Para un análisis más detallado de la prueba, por favor referirse al documento de Microsoft Excel anexo al proyecto.

### Prueba 3 – Escalabilidad



La latencia máxima al recibir la información de 4000 los sensores cuando se requiere actualizar el boletín de alerta fue de 97 milisegundos, inferior al tiempo de respuesta máximo de 1 minuto cuando se requiere actualizar el boletín de alerta. El error fue de 0.00% en todos los casos evaluados, por lo que concluimos que el sistema cumple con el requisito solicitado. Deducimos de esto la arquitectura elegida y las decisiones realizadas durante el desarrollo de cada una de sus partes fueron un factor determinante para el buen desempeño del servicio en cuestión. No obstante, de forma similar a la prueba 2, debemos tener en cuenta que la actualización de boletín de alerta requiere de acceso a la base de datos, aspecto que puede ser fatal conforme el estrés de la aplicación incrementa.

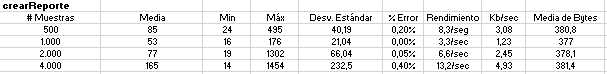
A continuación se puede observar el análisis gráfico que fue extraído a partir de esta prueba.

Para un análisis más detallado de la prueba, por favor referirse al documento de Microsoft Excel anexo al proyecto.

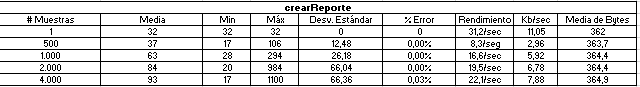
# Comparación resultados obtenidos Experimento 2 vs Experimento 1

## Prueba 1 – Generación de boletín de alerta:

Primero expondremos los resultados de las pruebas obtenidas en el experimento pasado:



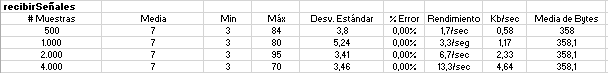
Y ahora volveremos a observar los resultados que obtuvimos en este experimento:



Al constrastar los resultados anteriormente expuestos podemos observar que claramente el grupo obtuvo, en primer lugar una mejora sustancial en el tiempo de respuesta medio que tardan en ser solucionadas las solicitudes de generar un reporte. Eso inicialmente, y por otro lado podemos ver que ahora en la gran mayoría de pruebas no nos encontramos con errores, de hecho, solo se presentan errores al ejecutarse 4.000 hilos de ejecución concurrentes y con apenas una tasa de error irrisoria del 0,03%. Es decir, gracias a la adición del balanceador de carga ahora nuestra aplicación es considerablemente mucho más escalable y por supuesto presenta una menor tasa de errores, por tanto la nueva arquitectura y componentes que presentamos hacen de nuestra aplicación actual una mejor alternativa que la que fue presentada en la entrega anterior.

## Prueba 2 – Recepción información y respuesta a cambios de nivel de altura

Primero expondremos los resultados de las pruebas obtenidas en el experimento pasado:



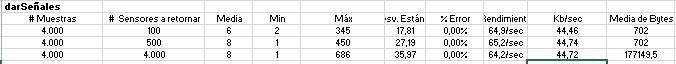
Y ahora volveremos a observar los resultados que obtuvimos en este experimento:



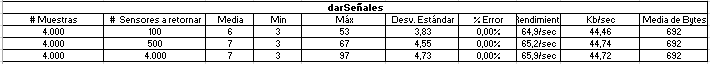
Al observar y comparar los resultados podemos decir que lo más claro es que reducimos el tiempo máximo de respuesta para la respuesta de una petición de recibir señales y generar reporte al presentarse un cambio en la altura de la ola. Es cierto que el tiempo de respuesta medio de las solicitudes aumentó, sin embargo hasta ahora sigue siendo más que satisfactorio para cumplir los escenarios de calidad propuestos por el enunciado por tanto es un factor que no debe ser tenido muy en cuenta.

## Prueba 3 – Escalabilidad:

Primero expondremos los resultados de las pruebas obtenidas en el experimento pasado:



Y ahora volveremos a observar los resultados que obtuvimos en este experimento:



Al comparar los resultados obtenidos en ambas versiones del sistema podemos afirmar con claridad que en esta entrega del sistema hemos obtenido un tiempo de respuesta para dar la información de todos los sensores en el borde costero gigantescamente superior en comparación con la de la entrega pasada, en esta ocasión hemos reducido a casi un sexto el tiempo de respuesta que estábamos presentando en la entrega pasada, por tanto esta versión del sistema es muchísimo mejor tanto a nivel de desempeño como a nivel de escalabilidad en comparación con la entrega pasada.

# Conclusiones

Para finalizar, tras la realización de las pruebas y el análisis de los datos obtenidos a partir de las mismas, observamos que la latencia máxima obtenida para la Prueba 1 en la etapa más exigente fue mayor a la máxima propuesta de 500 milisegundos, pero que en aquellas etapas de la prueba similares a situaciones reales el desempeño estuvo por debajo de este límite. Para la Prueba 2, observamos que el desempeño de la aplicación era satisfactorio en todos los contextos a los que fue sometida, siendo capaz de responder a los cambios del nivel de altura con una latencia máxima de 70 milisegundos. Para la Prueba 3, de manera análoga a la Prueba 2, observamos que incluso en la etapa más exigente, el sistema escaló de manera exitosa. Por lo anterior mencionado, concluimos que el sistema desarrollado es capaz de responder satisfactoriamente a todos los contextos a los que se verá sometido en la práctica.

Sin embargo, dada la naturaleza de los servicios que fueron testeados, concluimos también que debemos tener presente en todo momento que requieren de la realización de transacciones en la base de datos, de modo que mantengamos dichos accesos implementados de la forma más eficiente posible en todo momento para evitar un incremento en el tiempo de respuesta general de la aplicación.

También es menester mencionar que a partir de los argumentos expuestos en la sección de comparación de los resultados obtenidos en la entrega anterior con relación a esta nueva entrega podemos decir con seguridad que hemos obtenido una versión de nuestro sistema que es claramente superior al anterior y que será de gran ayuda en el futuro y que permitirá que el usuario final se sienta satisfecho con nuestra solución propuesta pues esta es capaz de cumplir de forma efectiva todos y cada uno de los escenarios de calidad propuestos.

Adicionalmente se debe mencionar que el grupo de trabajo deberá observar cuidadosamente los cambios que realice en la arquitectura con el fin de obtener mejores resultados en los distintos escenarios de calidad pues se nota una diferencia considerable en las distintas pruebas luego de añadir los elementos de persistencia e interfaz gráfica que terminaron teniendo un impacto más grande (en algunos requerimientos) del esperado.