**Documentación Entrega Parcial - Experimento 2**

**Grupo: 10**

**Pre-experimentación:**

1. **Problemática:** Hipótesis:El experimento es realizado en JAX-RS con una capa de servicios REST que soporta 1000 peticiones con información de los 3 sensores y tiene un tiempo de respuesta menor a 1000 ms, además se busca garantizar que el sistema cumpla con el atributo de calidad de disponibilidad, integridad de datos y seguridad.
2. **Objetivo:** El objetivo de este experimento es validar las decisiones de diseño que el equipo de trabajo ha tomado con respecto a los requerimientos funcionales y no funcionales.
3. **Descripción del experimento:** El experimento consiste en una prueba de carga para determinar la cantidad de usuarios simultáneos que soporta el sistema implementado con un balanceador de carga; esto es implementado con dos máquinas que exponen los servicios del sistema. Para ello, se utilizará JMeter como generador de carga.

Se probará el método post en la ruta:

[**http://localhost:8080/Servidor-JAXRS/api/alertas**](http://localhost:8080/Servidor-JAXRS/api/alertas)y se le pasará una alerta en formato JSON:

**{**

"esEmergencia"**:** **true,**

"fecha"**:** "2017-02-14T19:39:36.735-05:00"**,**

"frecuenciaCardica"**:** 10**,**

"idDispositivo"**:** 1**,**

"nivelEstres"**:** 10**,**

"presionSanguinea"**:** **[**

5**,**

8

**],**

"tipo"**:** 2**,**

"ubicacion"**:** **[**

5**,**

8

**]**

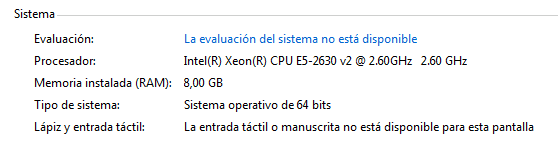
**}**

**Artefactos utilizados:** Capa de rest, capa de logic y capa de mocks.

1. **Recursos de la experimentación:**

Los recursos descritos a continuación son iguales tanto en las máquinas que prestan los servicios del sistema y las del balanceador de carga.

Hardware:



Software:

Windows 7, JDK8, Glashfish 4.1.0.

1. **Resultados esperados:** Se espera que el experimento supere las pruebas y que la hipótesis sea verdadera.
2. **Duración y etapas:** Se realizará la prueba de carga desde 100 threads hasta 1000 con intervalos de 100 threads; estos simulan la recepción de los datos que envían los brazaletes, finalmente se comprobará con 2000, 3000 y 4000 si las condiciones del experimento lo permiten.

**Post-experimentación**

1. **Resultados obtenidos:**

A continuación, se muestran los resultados obtenidos en la entrega parcial del experimento 2, esta entrega consiste en implementar un balanceador de carga para impactar directamente en el atributo de calidad de disponibilidad.

Como se puede evidenciar en la gráfica anterior, la implementación del balanceador de carga no afecto el cumplimiento del requerimiento principal del sistema (atender 1000 peticiones en menos de 1 segundo ), después de generar las pruebas y registrar los tiempos obtenidos, se puede evidenciar que se presenta una alta disponibilidad, ya que atender las mismas 1000 peticiones de los usuarios se responden en 326 ms lo que corresponde aproximadamente a un 70% menos del límite propuesto por el negocio, es decir, el sistema es realmente eficiente para los objetivos del hospital.

La especificación de los datos obtenidos se muestra a continuación: Tabla de resultado JMeter.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Columna1** | **# muestras** | **Media** | **Min** | **Max** | **Desv. estándar** | **error** |
| Petición HTTP | 100 | 29 | 19 | 215 | 23885007850113800 | 0.0 |
| Petición HTTP | 200 | 31 | 21 | 143 | 10945061671822500 | 0.0 |
| Petición HTTP | 300 | 41 | 28 | 292 | 16934771592462900 | 0.0 |
| Petición HTTP | 400 | 61 | 41 | 312 | 17658764820904000 | 0.0 |
| Petición HTTP | 500 | 81 | 61 | 132 | 10071154055022700 | 0.0 |
| Petición HTTP | 600 | 110 | 87 | 156 | 1228602175464280 | 0.0 |
| Petición HTTP | 700 | 142 | 116 | 215 | 13683368349487500 | 0.0 |
| Petición HTTP | 800 | 170 | 127 | 364 | 6855623139062100 | 0.0 |
| Petición HTTP | 900 | 120 | 166 | 247 | 12965405251411700 | 0.0 |
| Petición HTTP | 1000 | 197 | 85 | 319 | 18288463576801600 | 0.0 |

Ahora, se muestran los resultados obtenidos en la entrega final del experimento 2, esta entrega consiste en implementar seguridad en el sistema para impactar directamente en el atributo de calidad de confidencialidad e integridad.

Como se puede evidenciar en la gráfica anterior, la implementación de la seguridad no afecto el cumplimiento del requerimiento principal del sistema (atender 1000 peticiones en menos de 1 segundo ), después de generar las pruebas y registrar los tiempos obtenidos, se puede evidenciar una disponibilidad aceptable para el negocio. La especificación de los datos obtenidos se muestra a continuación: Tabla de resultado JMeter.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Columna1** | **# muestras** | **Media** | **Min** | **Max** | **Desv. estándar** | **error** |
| Petición HTTP | 100 | 100 | 29 | 115 | 23885007850113800 | 0.0 |
| Petición HTTP | 200 | 126 | 31 | 143 | 10945061671822500 | 0.0 |
| Petición HTTP | 300 | 132 | 48 | 292 | 16934771592462900 | 0.0 |
| Petición HTTP | 400 | 139 | 51 | 312 | 17658764820904000 | 0.0 |
| Petición HTTP | 500 | 173 | 61 | 132 | 10071154055022700 | 0.0 |
| Petición HTTP | 600 | 190 | 87 | 156 | 1228602175464280 | 0.0 |
| Petición HTTP | 700 | 221 | 116 | 215 | 13683368349487500 | 0.0 |
| Petición HTTP | 800 | 257 | 127 | 364 | 6855623139062100 | 0.0 |
| Petición HTTP | 190 | 292 | 166 | 247 | 12965405251411700 | 0.0 |
| Petición HTTP | 1000 | 309 | 185 | 319 | 18288463576801600 | 0.0 |

Finalmente, la siguiente grafica compara la eficiencia de la entrega parcial del experimento 2 con respecto a la entrega final. Como era de esperarse la eficiencia disminuyo aproximadamente un 25%, en este punto el sistema tiene un tiempo de respuesta para mil peticiones de 309ms, esto se debe a la implementación de la seguridad a pesar que se cuenta con un balanceador de carga.

A manera de conclusión como grupo deducimos que:

1. **Duración Real:**

El tiempo teórico o esperado es de 1000ms o menos preferiblemente, la gráfica anterior muestra que nuestros datos oscilan en un 70% menos, es decir, nuestra muestra registra una eficiencia de 309ms; esto hace que sea una solución efectiva y rígida si aumenta el número de solicitudes.

1. **Artefactos construidos:**

Los artefactos construidos para dar solución al problema para la recepción de los datos que generan los sensores fue una arquitectura por capas, estas son: una capa lógica, una capa de recursos, una capa de mocks, una base de datos relacional que garantiza la persistencia de los datos de la aplicación y finalmente un balanceador de carga que aumenta la disponibilidad del sistema para el negocio, pero es castigada con la implementación de la seguridad.

1. **Análisis:**

Los resultados obtenidos son satisfactorios, obedecen a la arquitectura desarrollada y permiten justificar la escogencia de la misma sobre las demás. Esto se sustenta por la eficiencia registrada en las pruebas de carga en JMeter la cual muestra una respuesta de 309ms para la petición de registro de 1000 clientes potenciales.

1. **Conclusiones:**

Con base en los resultados de las pruebas podemos concluir que las decisiones arquitectónicas tomadas fueron acertadas, el experimento dos final cumple con los atributos de calidad en un tercio del tiempo requerido. Finalmente se tiene claro los requerimientos solicitados y se cumple efectivamente con la solución propuesta y en un tiempo de respuesta óptimo.

Además se comparan las ventajas y desventajas de la última decisión de implementación tomada por el grupo.

**Descripción de la Arquitectura**

Con base en el laboratorio de escalabilidad y desempeño decidimos utilizar JAX-RS porque en términos de procesamiento de peticiones fue más ágil. Además, implementamos beans sin estado con el objetivo de favorecer la velocidad del sistema y procesar las 1000 peticiones en menos de un segundo; esto se debe a que el sistema no se bloquea si una petición no responde rápidamente por parte del cliente y puede seguir procesando más solicitudes al tiempo.

El sistema cuenta con una base de datos relacional provisionada por Amazon, la cual garantiza la persistencia de los datos del sistema y a su vez la duplicación de la misma, esta fue creada con su mayor normalización, es decir, un 70% de las relaciones creadas están en tercera forma normal.

Finalmente se implementó un balanceador de carga con el objetivo de aumentar la disponibilidad del sistema, pero esta ganancia fue castigada con la implementación de la seguridad, ya que reduce el desempeño, la modificabilidad, la usabilidad y la disponibilidad del sistema.