Documentación Experimento 1 entrega definitiva
Grupo: 10

Pre-experimentación:

1. **Problemática:** Hipótesis: el experimento se 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.

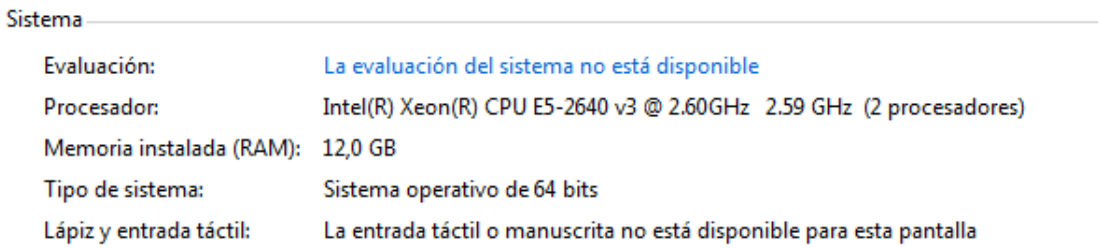
2. **Objetivo:** Validar la hipótesis o dar luz para tomar decisiones arquitectónicas.

3. **Descripción del experimento:** El experimento consiste en una prueba de carga para determinar la cantidad de usuarios simultáneos que soporta el 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** y se le pasará una alerta en formato JSON como se muestra a continuación:


```
{
 "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.

4. **Recursos de la experimentación:**
   Hardware:



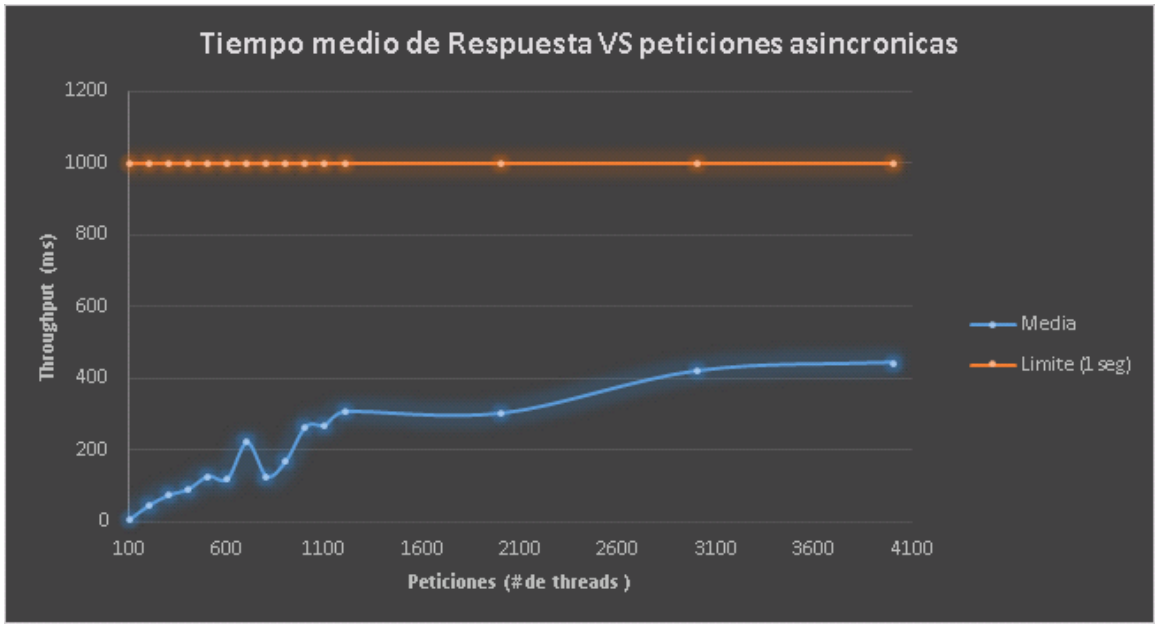
   Software:
   Windows 7, JDK8, Glashfish 4.1.0.
   Resultados esperados: Se espera que el experimento supere las pruebas y que la hipótesis sea verdadera.

5. **Duración y etapas:** Se realizará la prueba de carga desde 100 threads hasta 1000 con intervalos de 1000 y finalmente se comprobará con 2000, 3000 y 4000.

Post-experimentación

1. Resultados obtenidos:

Los resultados obtenidos en la entrega parcial de la implementación de la lógica del servidor fueron los siguientes: El sistema soporta la recepción de 1000 clientes potenciales los cuales cada uno envía en conjunto la muestra de 3 sensores, el tiempo promedio de respuesta es 266 ms. A continuación, se adjunta mediciones de la aplicación JMeter la cual permitió realizar la prueba de carga cuya grafica contrasta la eficiencia del producto generado:

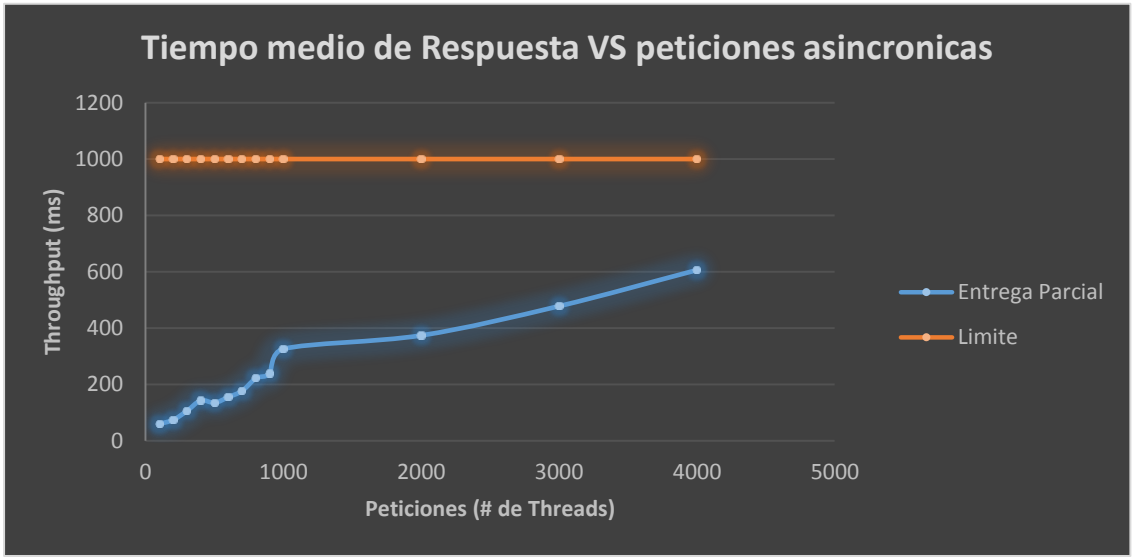


Como se puede se puede observar el producto generado cumple con el atributo de calidad especificado (eficiencia menor a 1000 ms) e incluso si debe contestar más de las peticiones solicitadas.
La especificación de los datos obtenidos se muestra a continuación: Tabla de resultado JMeter.

Columna1	# muestras	Media	Min	Max	Desv. estándar	error
Petición HTTP	200	8	2	84	13.259.622.920.731.900	0.0
Petición HTTP	200	48	2	363	6.704.755.308.734.240	0.0
Petición HTTP	300	76	1	561	7.882.886.076.107.460	0.0
Petición HTTP	400	92	1	440	10.288.025.758.132.600	0.0
Petición HTTP	500	128	1	1764	4.726.496.359.884.340	0.0
Petición HTTP	600	121	2	1894	44.988.234.510.999.200	0.0

Petición HTTP	700	222	1	1469	33.568.850.981.898.300	0.0
Petición HTTP	800	128	1	1762	4.241.812.683.275.860	0.0
Petición HTTP	900	172	1	2018	5.853.429.375.783.390	0.0
Petición HTTP	1000	266	1	1182	2.247.441.597.105.470	0.0

Los resultados obtenidos en la entrega definitiva de la implementación de la lógica del servidor y la persistencia de los datos fueron los siguientes: El sistema soporta la recepción de 1000 clientes potenciales los cuales cada uno envía en conjunto la muestra de 3 sensores y los persiste en una base de datos relacional, el tiempo promedio de respuesta es 326 ms. A continuación, se adjunta mediciones de la aplicación JMeter la cual permitió realizar la prueba de carga cuya grafica contrasta la eficiencia del producto generado para la primera parte del proyecto:



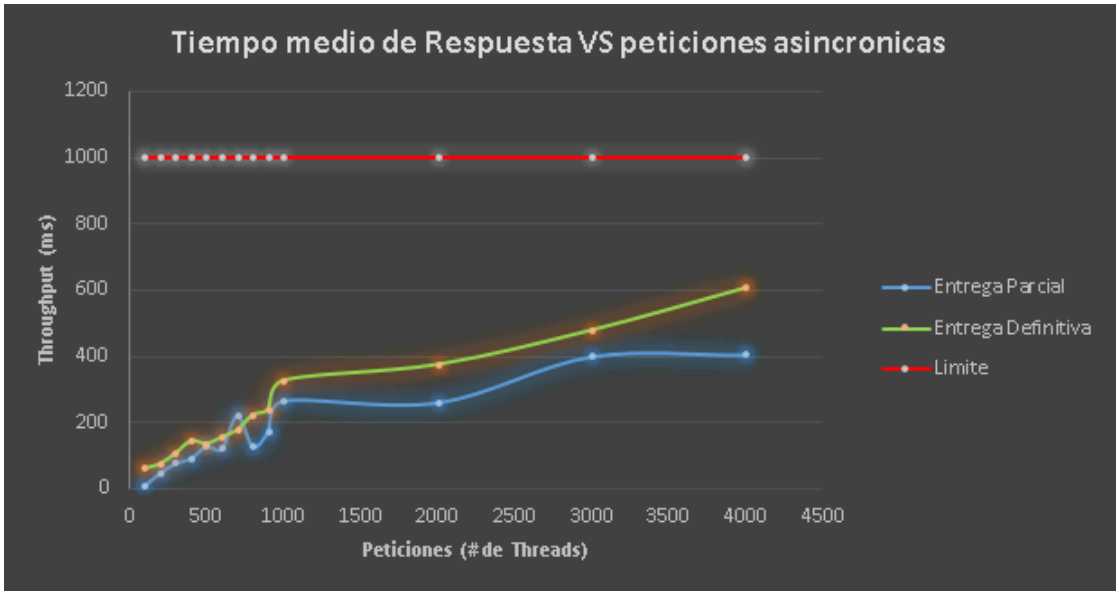
Como se puede se puede observar el producto generado con la persistencia aun cumple con el atributo de calidad especificado (eficiencia menor a 1000 ms) e incluso si debe contestar más de las peticiones solicitadas responde con buena eficiencia.

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

Columna1	# muestras	Media	Min	Max	Desv. estándar	error
Petición HTTP	200	60	2	84	13.259.622.920.731.900	0.0
Petición HTTP	200	74	2	363	6.704.755.308.734.240	0.0
Petición HTTP	300	106	1	561	7.882.886.076.107.460	0.0
Petición HTTP	400	143	1	440	10.288.025.758.132.600	0.0
Petición HTTP	500	135	1	1764	4.726.496.359.884.340	0.0
Petición	600	155	2	1894	44.988.234.510.999.200	0.0

HTTP						
Petición HTTP	700	177	1	1469	33.568.850.981.898.300	0.0
Petición HTTP	800	222	1	1762	4.241.812.683.275.860	0.0
Petición HTTP	900	239	1	2018	5.853.429.375.783.390	0.0
Petición HTTP	1000	326	1	1182	2.247.441.597.105.470	0.0

Finalmente, la siguiente grafica compara la eficiencia de la entrega parcial con respecto a la entrega definitiva. Como era de esperarse la eficiencia disminuyo en aproximadamente 45 ms, es decir, la aplicación tarda aproximadamente 45 ms más en responder las mismas peticiones que en la entrega parcial, el incremento de tiempo se debe a la implementación de una base de datos relacional, la cual nos permite persistir los datos de la aplicación; ahora, como los datos no se guardan en memoria principal de la maquina sino persisten en una base de datos, la aplicación debe hacer nuevos llamados, lo que significa que esta implementación es una capa más para el proyecto influyendo de manera negativa en el atributo de calidad de eficiencia pero positivamente en el atributo de integridad de datos.



2. Duración Real:

El tiempo teórico o esperado es de 1000 ms o menos preferiblemente, la gráfica anterior muestra que la solución definitiva para la primera parte del proyecto dada por nuestro grupo, oscila aproximadamente en un 70% menos al esperado, es decir, nuestra solución registra una eficiencia de 326 ms; esto hace que sea una solución efectiva y resistente si aumenta el número de solicitudes.

3. 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 y finalmente una capa de persistencia usando una base de datos relacional llamada Derby.

4. **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 326 ms para la petición de registro de 1000 clientes potenciales.

5. **Conclusiones:**

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

Descripción de la Arquitectura

Con base en el laboratorio de escalabilidad y desempeño decidimos utilizar JAX-RS porque en términos de procesamiento de peticiones POST 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, finalmente implementamos una base de datos relacional llama Derby para garantizar la persistencia de los datos en la aplicación; esto influyó negativamente en nuestro atributo de calidad de eficiencia pero afecto positivamente el atributo de integridad de datos, aun así la aplicación desarrollada cumple con la restricción de eficiencia.