**Desempeño**

**Pre-experimentación**

1. **Problemática:** Se desea conocer si la latencia de respuesta a un pedido REST por parte del servidor en la arquitectura desarrollada es menor a 1 segundo.
2. **Objetivo del experimento:** El objetivo es poner a prueba la hipótesis con cada servicio REST presente en el software desarrollado.
3. **Descripción del experimento:** Se realizarán solicitudes REST a la aplicación por medio de JMeter y se analizará el tiempo de espera medio de los resultados para determinar si el software cumple con la hipótesis.
4. **Artefactos a construir:** Back-end de la aplicación.
5. **Recursos de la experimentación:** Máquina virtual en donde se ejecutarán las pruebas y se montará el servidor de la aplicación, Back-end de la aplicación y JMeter.
6. **Resultados esperado:** Se espera que el tiempo de respuesta promedio de las peticiones REST sea menor o igual a 1 segundo.
7. **Duración y etapas:** Se plantea construir el back-end de la aplicación basado en una arquitectura sincrónica que hace uso del API JAX-RS. Se realizan pruebas sobre esta arquitectura. Si las pruebas resultan satisfactorias se conserva esta arquitectura. Si las pruebas resultan insatisfactorias se cambiará la arquitectura a una asincrónica basada en el Play! Framework.

**Post-experimentación**

1. Resultados obtenidos
2. Duración real: Realizar una comparación con los tiempos teóricos y los reales de desarrollo para cada etapa del proceso de experimentación.
3. Artefactos construidos: Cuáles artefactos fueron construidos y cuáles no, explicando las causas por las que no fueron desarollados.
4. Análisis: Interpreta, explica y justifica los resultados obtenidos basado en patrones y tácticas arquitecturales implementadas en el experimento.
5. Conclusiones: Teniendo en cuenta la hipótesis, plantea recomendaciones basadas en los resultados obtenidos en la etapa de experimentación.

**Escalabilidad**

1. **Problemática:** Se desea conocer si el sistema capaz de responder y aguantar 4500 solicitudes REST en 5 segundos.
2. **Objetivo del experimento:** El objetivo es poner a prueba la hipótesis de que el software es capaz de resistir y responder 4500 solicitudes REST en 5 segundos.
3. **Descripción del experimento:** Se realizarán 4500 solicitudes REST a la aplicación por medio de JMeter y se analizará si el sistema presenta errores al procesarlas.
4. **Artefactos a construir:** Back-end de la aplicación.
5. **Recursos de la experimentación:** Máquina virtual en donde se ejecutarán las pruebas y se montará el servidor de la aplicación, Back-end de la aplicación y JMeter.
6. **Resultados esperado:** Se espera que el software responda a las 4500 solicitudes sin errores
7. **Duración y etapas:** Se plantea construir el back-end de la aplicación basado en una arquitectura sincrónica que hace uso del API JAX-RS. Se realizan pruebas sobre esta arquitectura. Si las pruebas resultan satisfactorias se conserva esta arquitectura. Si las pruebas resultan insatisfactorias se cambiará la arquitectura a una asincrónica basada en el Play! Framework.

**Post-experimentación**

1. Resultados obtenidos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Etiqueta | #muestras | media | min | max | Desv.Estandar | %error |
| readMovibusControllerPrueba | 250 | 512 | 85 | 2119 | 3.355.878.788.037.490 | 0.0 |
| createMovibusControllerPrueba | 250 | 4 | 3 | 17 | 10.001.679.858.903.700 | 0.0 |
| createUsuarioControllerPrueba | 2000 | 15 | 1 | 466 | 44.246.303.517.921.100 | 0.0 |
| getUsuarioControllerPrueba | 2000 | 3 | 1 | 110 | 6.667.058.646.809.700 | 0.0 |
| solicitarMovibusUsuarioControllerPrueba | 2250 | 14 | 2 | 246 | 26.997.027.585.812.700 | 0.0 |
| createConductorControllerPrueba | 1500 | 3 | 1 | 288 | 8.495.025.341.666.470 | 0.0 |
| readConductorControllerPrueba | 1000 | 137 | 8 | 726 | 15.464.542.372.472.500 | 0.0 |
| getConductorControllerPrueba | 1000 | 2 | 1 | 113 | 4.077.373.541.877.170 | 0.0 |
| createTranviaControllerPrueba | 250 | 3 | 1 | 26 | 16.588.236.795.994.900 | 0.0 |
| readTranviaControllerPrueba | 250 | 188 | 73 | 386 | 6.745.763.340.052.770 | 0.0 |
| getTranviaControllerPrueba | 250 | 3 | 3 | 42 | 2.553.432.200.000.610 | 0.0 |
| updateTranviaControllerPrueba | 250 | 5 | 4 | 32 | 20.358.546.117.048.700 | 0.0 |
| reportarAccidenteTranviaControllerPrueba | 250 | 3 | 3 | 21 | 12.770.215.346.657.200 | 0.0 |
| readReporteControllerPrueba | 500 | 7 | 5 | 83 | 6.560.613.081.107.580 | 0.0 |
| getReporteControllerPrueba | 500 | 4 | 1 | 68 | 5.363.137.514.552.460 | 1.0 |
| createEstacionVcubControllerPrueba | 200 | 4 | 3 | 44 | 29.898.787.600.837.500 | 0.0 |
| readEstacionVcubControllerPrueba | 200 | 5 | 3 | 23 | 16.348.623.795.292.300 | 0.0 |
| llenarEstacionVcubControllerPrueba | 200 | 4 | 1 | 20 | 1.380.208.317.610.060 | 0.0 |
| retirarVcubDeEstacionVcubControllerPrueba | 200 | 4 | 4 | 27 | 16.983.521.425.193.300 | 0.0 |
| devoulcionVcubEstacionVcubControllerPrueba | 200 | 4 | 4 | 11 | 9.420.721.840.708.380 | 0.0 |

1. Duración real: Realizar una comparación con los tiempos teóricos y los reales de desarrollo para cada etapa del proceso de experimentación. Todos los resultados sacaron muy buenos resultados, pero en las primeras etapas de prueba, usualmente la primera vez que lo corríamos, se demoraba mucho más de lo esperado ya que estaba subiendo los recursos al servidor y esto causaba un gran aumento de tiempo de ejecución en el despliegue. En general los resultados fueron mucho mejor de los esperados después del despliegue y la primera ejecución de prueba de los datos.
2. Artefactos construidos: Cuáles artefactos fueron construidos y cuáles no, explicando las causas por las que no fueron desarrollados. Desarrollamos toda la arquitectura basados en un modelo de actores teniendo en cuenta que el software tenga una buena escalabilidad y desempeño. Las métricas que se propusieron al inicio del proyecto fueron cumplidas gracias al uso de framework play y se descartó el uso de Jaxrs ya que al ser asincrónico lo hace más lento y no cumple con la escalabilidad ya que no tiene las ventajas que tiene play con los actores.
3. Análisis: Interpreta, explica y justifica los resultados obtenidos basado en patrones y tácticas arquitecturales implementadas en el experimento.

Nosotros implementamos una arquitectura asincrónica basada en actores lo que genera que los actores solo sean creados cuando sea necesario lo que genera una eficiencia más alta y nos permite cumplir con los requerimientos de escalabilidad y desempeño.

1. Conclusiones: Teniendo en cuenta la hipótesis, plantea recomendaciones basadas en los resultados obtenidos en la etapa de experimentación.

Se tiene que es mejor el uso de play para el manejo del problema con un enfoque hacia el desempeño y escalabilidad ya que esta provee los medios para un mejor manejo de una mayor cantidad de usuarios en comparación con Jaxrs. Las pruebas de estrés no solo nos permitieron comprobar las métricas propuestas al inicio del proyecto, sino también verificaron funcionalidades adicionales que aumentan el valor del negocio.