Javier Esteban Camargo 20131345

David Felipe Sabogal 201327594

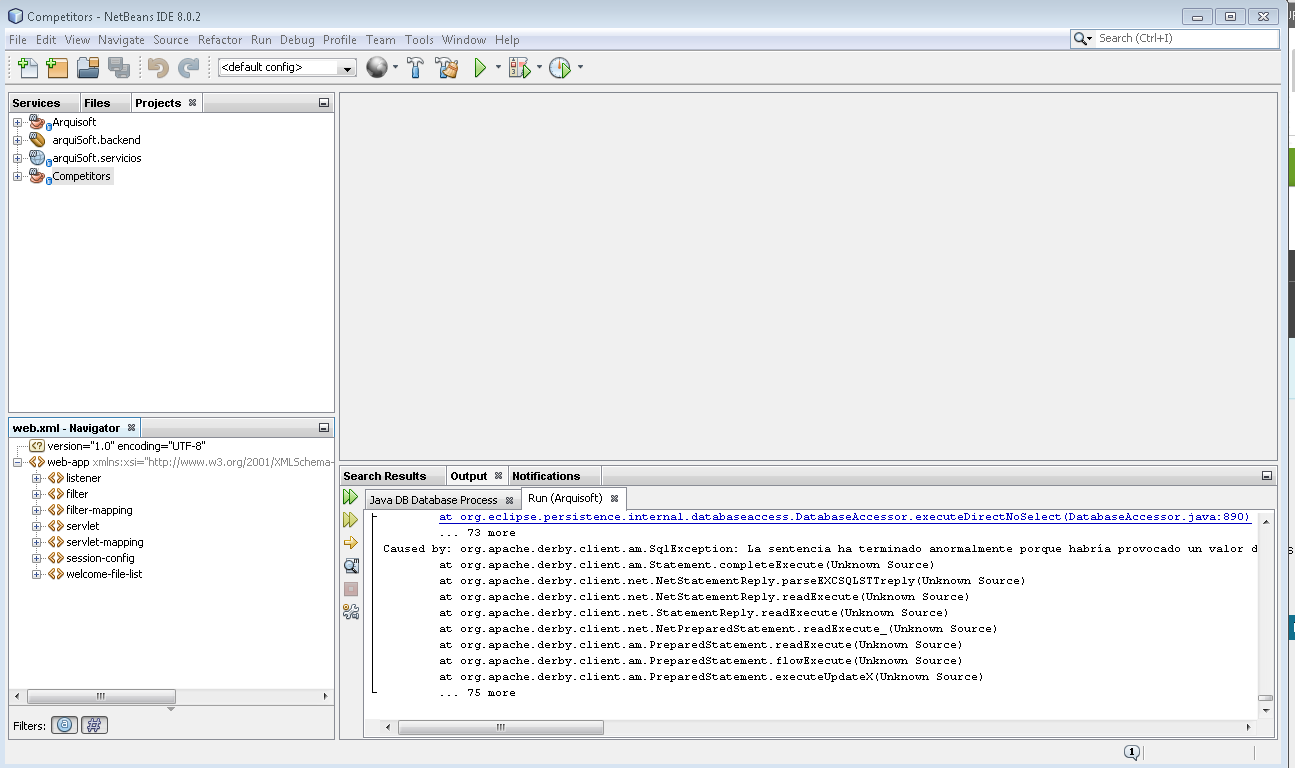
Jose Alejandro Silva 201312889

David Camilo Bonilla 201313347

Experimento 2

**Disponibilidad**

Para el atributo de calidad de disponibilidad, se crearon tres instancias de la aplicación distribuidas en diferentes locaciones, donde cada instancia está en capacidad de responder a cabalidad las diferentes solicitudes hechas a la aplicaciones, cada uno sirve de respaldo al momento de que cualquier de los otros servidores presente algún tipo de fallas, se configuró el sistema de modo que en caso de que los dos servidores fallen o no se encuentren en línea, siempre haya uno activo. Adicionalmente, cada servidor cuenta con la versión actualizada de la aplicación, para mantener el despliegue activo constantemente y ayudar a responder varias solicitudes, en el siguiente ejemplo se observa una demostración del sistema para garantizar la disponibilidad



Acá se observa que los nodos 2 y 3 han fallado la ejecución por algún motivo, entonces ninguno de estos es capaz de atender solicitudes, por consiguiente estos nodos no se encuentra disponible. A pesar de lo anterior, la aplicación tiene el respaldo de un tercer servidor para estos casos, entonces acá observamos la respuesta a una solicitud, con un tiempo mucho mayor a lo acostumbrado, pero garantizando la disponibilidad de la aplicación. Gracias al sistema anterior, se puede cumplir el cumplir el escenario de calidad de estar disponible 365 días al año, ya que siempre va a haber un servidor en línea mientras se arreglan los problemas en los demás, adicionalmente existe el respaldo en la web para casos extremos

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| darUsuarios | 10000 | 394 | 36 | 918 | 273.9577205 | 0 | 565.6851642 | 107.723249 | 195 |

**Pruebas**

Con respecto a la eficiencia del programa, dado que se está utilizando un balanceador de carga, es importante determinar qué política es la más eficiente para la aplicación, se probaron dos diferentes políticas con el mismo tipo de solicitudes para comprobar cual es mejor para la aplicación, estos son los resultados..

**Round-Robin**

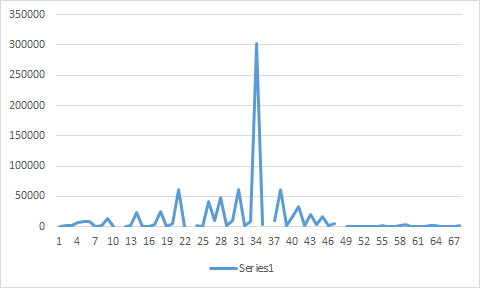
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Etiqueta | #Muestras | Media | Mín | Max | Desv. Estándar | %Error | Rendimiento | Kb/sec | Media de Bytes |
| darUsuarios | 1000 | 233 | 9 | 922 | 253.9627745 | 0 | 663.1299735 | 244.140625 | 377 |
| darTranvias | 1000 | 480 | 12 | 1206 | 380.8810494 | 0 | 305.1571551 | 128.4414144 | 252 |
| darReservas | 1000 | 414 | 9 | 959 | 337.8087329 | 0 | 628.1407035 | 119.6166379 | 195 |

### **Least-connection**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Etiqueta | #Muestras | Media | Mín | Max | Desv. Estándar | %Error | Rendimiento | Kb/sec | Media de Bytes |
| darUsuarios | 1000 | 277 | 20 | 808 | 211.77145 | 0 | 539.3743258 | 102.7128843 | 195 |
| darTranvias | 1000 | 279 | 6 | 1008 | 288.3687333 | 0 | 521.9206689 | 58.11098184 | 195 |
| darReservas | 1000 | 311 | 20 | 1088 | 304.1647118 | 0 | 571.1022273 | 108.7548187 | 195 |

Del resultado anterior es posible determinar que en la gran mayoría de pruebas la política del balanceador de least-connection es mucho más eficiente en el contexto de la aplicación, principalmente por que no se tiene manejo completo de los servidores externos donde se está corriendo la aplicación, esto puede ocasionar que estos una petición sea demorada ya que no hay control de los recursos computacionales. Por lo anterior, es mejor balancear la carga de acuerdo al menor número de conexión activas por servidor, soportado en los resultado el rendimiento es mucho mayor y la cantidad de información referente a las solicitudes es mucho menor.

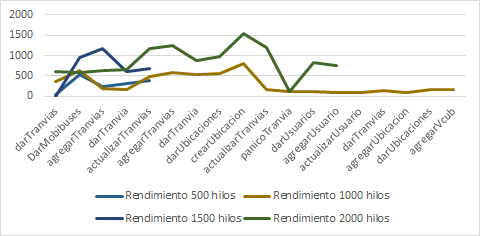
**Requerimientos y Latencia**



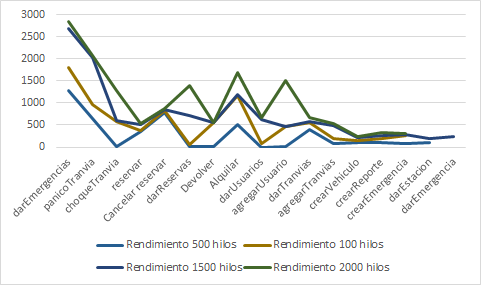
Observando esta gráfica de requerimientos cada uno con su latencia, en la cual los picos significan las subidas de latencia dependiendo de los hilos con los que se realizó la prueba, podemos observar que hay requerimientos donde la pendiente del pico es considerablemente mayor a la de otros requerimientos mientra que la pendiente de otros es casi 0. Esto se da debido a que hay requerimientos donde se deben realizar muchas más operaciones contra la base de datos las cuales son costosas para el servidor y aumentan la latencia notablemente. Por ejemplo, la operacion de dar todas las emergencias las cuales están relacionadas con vehiculos y ubicaciones es una operacion costosa ya que debe retornar gran cantidad de datos tanto de la tabla de emergencias como de la de ubicaciones y la de vehículos, la cual tambien tiene relacion con reportes, etc. Otros requerimientos en los que podemos observar una gran latencia fueron agregar usuario, agregar reserva y alquiler de Vcubs ya que también involucran operaciones con una gran cantidad de entidades.

**Rendimiento**

**Anteriormente**



**Ahora**



Podemos muchos de los requerimientos el rendimiento ha aumentado ya que ahora se dispone de un balanceador de carga y un mayor número de servidores en los cuales se puede distribuir las solicitudes. Esto deriva en que se usa un mayor número de recursos computacionales ya que existen 3 servidores en 3 distintas máquinas virtuales por lo que podemos ver un aumento en el rendimiento de las solicitudes.