

MAVITY

Entrega 2.1

Contenido

[Restricciones 1](#_Toc443567782)

[Decisiones de Arquitectura 2](#_Toc443567783)

[Atributos de Calidad 3](#_Toc443567784)

[Resumen de métricas 4](#_Toc443567785)

Entrega 2

[Reporte de modificaciones arquitecturales](#_Toc443567787)

[Resumen datos de experimento 1](#_Toc443567787)

[Escenario de calidad Entrega 2](#_Toc443567787)

[Reporte de resultados Entrega 2](#_Toc443567787)

[Integrantes 10](#_Toc443567787)

[Vistas Arquitecturales 11](#_Toc443567786)

[Integrantes 12](#_Toc443567787)

[Mavity 13](#_Toc443567788)

# Restricciones

|  |  |  |  |
| --- | --- | --- | --- |
| Identificador | Tipo | Restricción | Establecida por |
| JEE para la aplicación | Tecnológica | Se debe usar este tipo de aplicación para poder desacoplar los servicios del negocio de la lógica y la interfaz. | Lineamientos del curso de Arquisoft. |
| Tiempo de clase | Gerencia-empresarial | Es necesario arquitecturar, diseñar y desarrollar la aplicación en un periodo de 3 meses. | Cronograma Uniandes. |
| Equipo de desarrollo reducido | Gerencia-empresarial | Se tiene un equipo de 4 personas con tiempo limitado. | Lineamientos del curso. |

# Decisiones arquitecturales

Para favorecer el cumplimiento de los escenarios de calidad de desempeño decidimos usar una arquitectura REST con 3 capas, estas capas son: interfaz, servicio y back-end, esta última se encarga de la parte lógica y la persistencia. Por otro lado, no fue necesario usar alguna táctica de desempeño debido a que el proyecto ya cumple con la latencia solicitada.

Por  otro lado, para favorecer el cumplimiento de los escenarios de calidad de rendimiento se usó una estilo arquitectural Pull, ya que se reciben las peticiones en una cola donde se tipifican y se priorizan. Además este estilo arquitectural disminuye el riesgo de pérdida de peticiones de la arquitectura REST.

# Atributos de Calidad

Atributo: DESEMPEÑO

Escenario de calidad Desempeño absoluto

|  |  |  |  |
| --- | --- | --- | --- |
| * Identificador | * Prioridad | | * Estimulo |
| Desempeño-Absoluto | Alta | | Envió de un dato del sensor |
| Fuente | | | |
| Sensores ubicados en los diferentes puntos de las costas. | | | |
| Ambiente | | Medida esperada | |
| Sensor envía dato. | | 500 ms por Thread-estimulo | |

Escenario de calidad Desempeño de escalabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| * Identificador | * Prioridad | | * Estimulo |
| Desempeño-Escalabilidad | Alta | | Envió de datos desde el sensor |
| Fuente | | | |
| Sensores ubicados en los diferentes puntos de las costas. | | | |
| Ambiente | | Medida esperada | |
| Sensores envían datos y sistema agrega estos mismos. | | 4000 entradas por minuto. | |

# Resumen de métricas

## Escenario de Calidad 1

|  |  |  |  |
| --- | --- | --- | --- |
| Escenario de Calidad | * Métrica | * Valor esperado | Valor Obtenido |
| Escenario 1 | Desempeño absoluto | 500ms | 0,333ms |

## Escenario de Calidad 2

|  |  |  |  |
| --- | --- | --- | --- |
| Escenario de Calidad | * Métrica | * Valor esperado | Valor Obtenido |
| Escenario 1 | Desempeño absoluto | 500ms | 0,333ms |

Relación entre escenarios de calidad Rendimiento absoluto vs Rendimiento de latencia

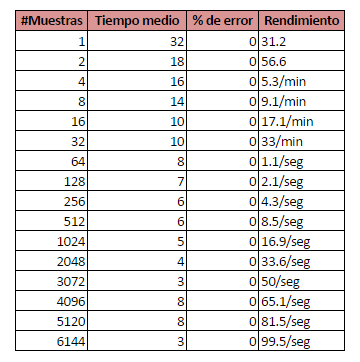
# ENTREGA 2

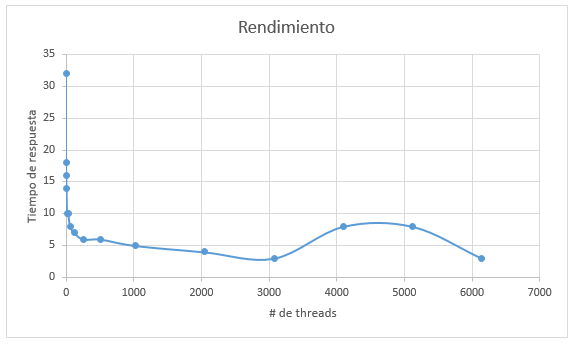
# Reporte de modificaciones arquitecturales y manejo del proyecto en github

Se decidió abrir una nueva rama del proyecto en la que se va a implementar la persistencia mediante unos cambios a la arquitectura, ya que no se va a hacer uso de JPA sino se quiere tener una conexión directa a una base de datos Oracle, de la cual se va a tener completo control. Esto con el objetivo de tener un mayor control de los datos que se necesitan hacer persistir y tener mayor control sobre el flujo de la información. Para esta nueva implementación, en la arquitectura se cambia la estructura de componentes y de flujo de la información: El servicio expuesto en la clase servicio, ahora pasa a llamar a un método en la clase Master, la cual maneja todas las transacciones y la lógica de los requerimientos que se deben exponer. En última instancia, esta clase Master llama a una clase DAO (por cada tabla en la base de datos hay un DAO), la cual contiene el método SQL que es ejecutado sobre la base de datos para hacer operaciones de lectura o escritura.

En esta nueva implementación, además se espera tener menor overhead, debido a que no se usa el API para la persistencia sino se maneja directamente.

# Resumen experimento





# Escenario de calidad entrega 2

Atributo: DESEMPEÑO

Escenario de calidad Desempeño absoluto

|  |  |  |  |
| --- | --- | --- | --- |
| * Identificador | * Prioridad | | * Estimulo |
| Desempeño-Absoluto | Alta | | Obtener un boletín |
| Fuente | | | |
| Boletín generado | | | |
| Ambiente | | Medida esperada | |
| Sistema envía el boletín | | 500 ms por Thread-estimulo | |

Escenario de calidad Desempeño de escalabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| * Identificador | * Prioridad | | * Estimulo |
| Desempeño-escalabilidad | Alta | | Envión de boletín desde el sistema |
| Fuente | | | |
| Sistema en la lógica genera un boletín y lo entrega desde un servicio | | | |
| Ambiente | | Medida esperada | |
| Sistema envían datos del boletín. | | 4000 entradas por minuto. | |

# Resumen de métricas

## Escenario de Calidad 1

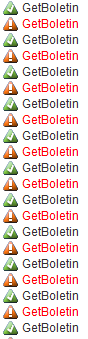
|  |  |  |  |
| --- | --- | --- | --- |
| #Muestras | Tiempo medio | % error | Rendimiento |
| 1 | 28 | 0 | 35.7/min |
| 2 | 16 | 50% | 4.0/min |
| 25 | 12 | 48% | 26.0/min |
| 75 | 11 | 50.67 | 1.3/min |
| 150 | 11 | 50% | 2.5/min |
| 300 | 11 | 50% | 5/min |
| 700 | 11 | 50% | 11.6/min |
| 1800 | 10 | 50% | 29.6% |
| 3500 | error |  |  |
| 4500 |  |  |  |
| 5100 |  |  |  |
| 6100 |  |  |  |

REPORTE DE RESULTADOS ENTREGA 2.1

Se obtienen resultados que no son del todo los esperados ya que los componentes incluidos en la arquitectura y las decisiones de diseño no incluidas en la arquitectura no incidieron positivamente en la misma. En primer lugar observamos que el rendimiento decayó considerablemente y se ve claramente que el balanceo de la carga que debería esta haciendo el balanceador no se esta efectuando. (Observar imagen 00.1)

Pensamos que el error debe ser a nivel de las conexiones de bajo nivel que se establecen entre las aplicaciones y el balanceador. En algunos momentos las maquinas fallas y esto genera que se vea afectado el funcionamiento del sistema. Esto nos lleva directamente a pensar que el balanceador de carga tiene alguna falla que puede ser a nivel de los nodos conectados a este. En segunda instancia, podemos pensar que la manera como estamos manejando la persistencia con JDBC puede afectar el rendimiento, aunque con certeza esta decisión de implementación no es la que esta afectando las pruebas más allá del rendimiento de las mismas. Concluimos resaltando que se deben hacer pruebas a nivel del balanceador de carga para saber en que punto de la arquitectura esta fallando el balance que se ejecuta desde este mismo en relación a los nodos y la infraestructura de la aplicación. Cuando llegamos a 3500 en el grupo de hilos, el jmeter no presenta respuesta y esto se debe al funcionamiento presentado por la app.

En ultimo caso, verificamos cual era el problema y después de monitorear el funcionamiento de los nodos, observamos que la maquina virtual de 172.24.100.33 presenta falla, a lo que intentamos responder con un mantenimiento local. Sobre este problema analizamos que efectivamente el balanceo de carga no se esta realizando adecuadamente porque sin importar las fallas locales de los nodos remotos el componente debería dirigir el servicio a otro nodo para respaldar la operación.

img. 00.1

# Integrantes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TATIANA bARBOSA Arquitectura |  | aLEJANDRO sALAMANCA Desarrolador |  | aNDRES mURILLO pRUEBAS y diseño |
|  |  |  |  |  |
|  |  |  |  |  |
| fELIPE cueto Desarrollador |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Mavity

**Arquitectura de Software**

**Uniandes**

2016