Documento de Testing
Grupo 18 - 2022

Taller de Sistemas de Información Java EE

Mauricio Iglesias

Manuel Biurrun

Mathias Fernández

Julio 2022

Tabla de Contenido

1. Introducción **3**
1.1. Objetivo del documento . . . 3
1.2. Especificaciones técnicas . . . 3

2. Escenarios **4**
2.1. Flujo: Proponer iniciativa . . . 4
2.2. Flujo: Obtener certificado de participación . . . 4
2.3. Flujo: Visualizar dashboard . . . 4

3. Estrategia **5**
3.1. Definición . . . 5
3.2. Metodología . . . 5
3.3. Se medirá . . . 5
3.4. Herramientas . . . 5

4. Resultados **6**
4.1. Coverage . . . 6
4.2. Prueba 1 . . . 6
4.3. Prueba 2 . . . 7
4.4. Prueba 3 . . . 8
4.5. Punto de quiebre . . . 9
4.6. Elastic cloud . . . 10

5. Optimizaciones **11**
5.1. Cuellos de botella . . . 11
5.2. Concurrencia . . . 11
5.3. Entorno JVM . . . 11
5.4. SQL . . . 11
5.5. MongoDB . . . 12

1. Introducción

El producto a ser probado es eParticpatation.uy. Las funcionalidades que se pondrán a prueba seleccionadas entre las más utilizadas del sistema, las más demandantes y las que tienen mayor complejidad.

1. **Proponer Iniciativa**
2. **Obtener certificado de participación**
3. **Visitar dashboard**

1.1. Objetivo del documento

- Se busca comprobar el desempeño de la plataforma «eparticipation.uy» frente a distintos escenarios de la realidad esperada.
- Mostrar el resultado de las pruebas unitarias, el cubrimiento logrado, pruebas de stress y rendimiento.
- En base a los resultados, analizar posibles mejoras de optimización del software.

1.2. Especificaciones técnicas

- **Se declara un ambiente de pre-producción, el cual el servidor correrá la aplicación final eparticipation.uy.**
- **Se declara un ambiente de testing, el cual el equipo de pruebas ejecutará los test unitarios referentes a eparticipation.uy y los escenarios propuestos; se trata de un clon del componente de negocio y datos, clon del driver jdbc:/postgresql, clon JVM SE 11.**

Servidor

- CPU: Intel Core i7 4930K @ 3.40GHz
- RAM: 24GB DDR3
- SSD: 1TB
- SO: Windows 10 64-bit

Equipo de pruebas

- CPU: Intel Core i7 4930K @ 3.40GHz
- RAM: 24GB DDR3
- SSD: 1TB
- SO: Windows 10 64-bit

Goals

- Verificar el comportamiento frente a situaciones de uso previsto.
- Hallar del punto de quiebre del sistema atravéz de pruebas de estress.
- Encontrar posibles 'Cuellos de botella' en la aplicación.
- Verificar escalabilidad y distribucion de carga en Elastic cloud.

2. Escenarios

- El alcance de estas prueas está limitado a determinados endpoints REST de la aplicación «eparticipation.uy», donde se evaluará la perfomance y los puntos de quiebre.
  - Proponer iniciativas: Múltiples usuarios y organizaciones simultáneamente.
  - Obtener certificado de participación: Múltiples usuarios obtienen certificados de participación.
  - Visualizar dashboard general: múltiples usuarios accederán a la página principal de «eparticipation.uy».

2.1. Flujo: Proponer iniciativa

- **POST** *eParticipation-periferico-organizacion/iniciativa/proponer*
- **POST** *eParticipation/iniciativa/alta*

2.2. Flujo: Obtener certificado de participación

- En particular, esta prueba tiene sus casos esperados en los multiplos de 10, es decir al intento 10, 20, 30, etc.
- **POST** *eParticipation/iniciativa/alta* → **Si es la decima(o múltiplo) creación de una iniciativa del usuario se le otorgará un certificado de participacion nivel 1(o su equivalente) por crear 10 inicitivas en «eparticipation.uy».**
- **GET** *eParctipation/usuario/listarCertificados*

2.3. Flujo: Visualizar dashboard

- **GET** *eParticipation.web.elasticloud.uy*
- **GET** *eParticipation/iniciativa/listar*
- **GET** *eParticipation/proceso/listar*

3. Estrategia

3.1. Definición

La técnica sera causar stress en los endpoints y medir los tiempos de respuesta. Las pruebas simularan una saturación del servicio de muchos usuarios consumiendo el sistema de forma concurrente.

3.2. Metodología

Se probarán los endpoint con una baseline de 1 usuario y varias iteraciones, incrementando por diez la cantidad de usuarios e iteraciones hasta lograr el punto de quiebre.

3.3. Se medirá

- Tiempo promedio de respuesta.
- El rendimiento (peticiones/min)

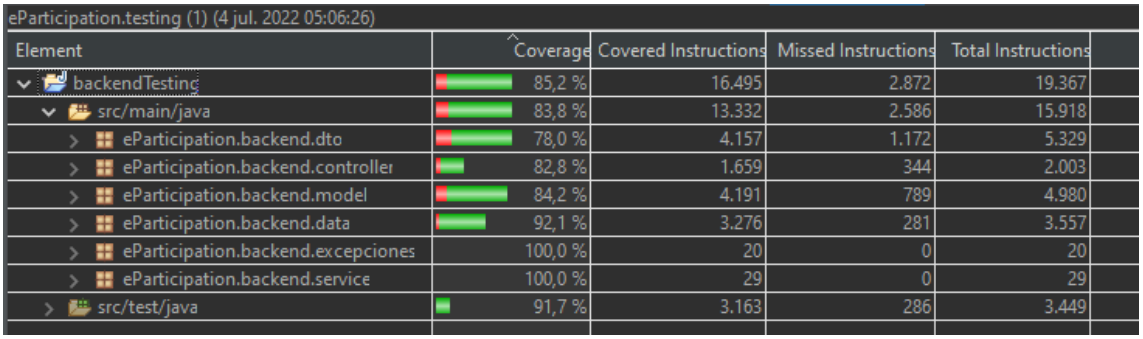
3.4. Herramientas

- JUnit con EclEmma - Testing [1][2]
- JMeter - Pruebas de carga y stress [3]
- Glowroot APM - Monitoreo [4]
- Consola Elastic-cloud - Monitoreo

4. Resultados

4.1. Coverage

Fueron realizados todos los test unitarios de la capa lógica de eparticipation.uy. Se realizaron pruebas para comprobar el funcionamiento de las distintas funcionalidades de la capa lógica y de datos. Con la herramienta JUnit 5 y el plug-in EclEmma se logró un cubrimiento de 85,2 %, que representa un estado funcional de la aplicación.



eParticipation.testing (1) (4 jul. 2022 05:06:26)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
backendTesting	85,2 %	16.495	2.872	19.367
src/main/java	83,8 %	13.332	2.586	15.918
eParticipation.backend.dto	78,0 %	4.157	1.172	5.329
eParticipation.backend.controller	82,8 %	1.659	344	2.003
eParticipation.backend.model	84,2 %	4.191	789	4.980
eParticipation.backend.data	92,1 %	3.276	281	3.557
eParticipation.backend.excepciones	100,0 %	20	0	20
eParticipation.backend.service	100,0 %	29	0	29
src/test/java	91,7 %	3.163	286	3.449

Figura 1: Cubrimiento de eparticipation.uy.

4.2. Prueba 1

- 1 usuario - 10 iteraciones [**10 muestras**]

Tabla 1: Resultados obtenidos flujo 1

Http Request	Path	Tiempo respuesta (ms)	Throughtput
POST	/alta/iniciativa(ciudadano)	82 ms	2,5 request/sec
POST	/alta/iniciativa(organizacion)	677 ms	1,3 request/sec

Para la siguiente prueba se utilizó al usuario ciudadano del sistema "colo@gmail" para crear 10 iniciativas, luego al décimo intento de obtener el certificado del user, se arroja el resultado esperado.

Tabla 2: Resultados obtenidos flujo 2

Http Request	Path	Tiempo respuesta (ms)	Throughtput
POST	/alta/iniciativa (ciudadano)	52 ms	6,7 request/sec
GET	/usuario/listarCertificados/{user}	82 ms	1,3 request/sec

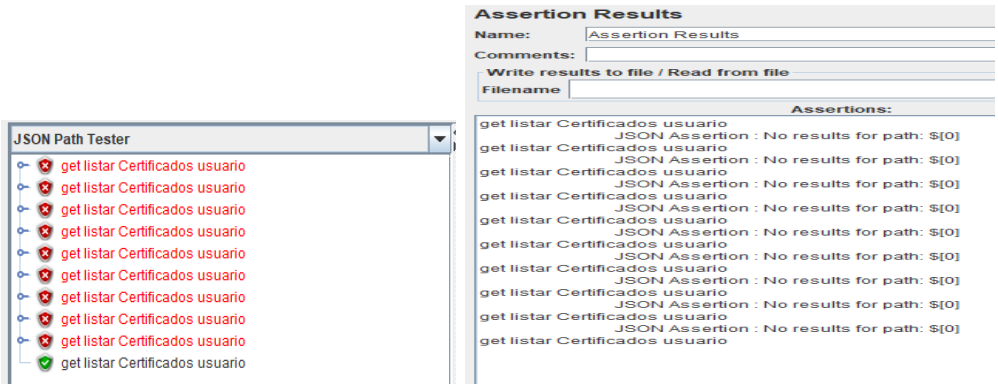


Figura 2: *Assertion result.*

Tabla 3: Resultados obtenidos flujo 3

Http Request	Path	Tiempo respuesta (ms)	Throughtput
GET	we-Participation.web.elasticloud.uy	26 ms	15,3 request/sec
GET	/iniciativa/listar	13 ms	15,6 request/sec
GET	/proceso/listar	14 ms	27,2 request/sec

4.3. Prueba 2

- 10 usuarios - 50 iteraciones [**500 muestras**]

Tabla 4: *Resultados obtenidos flujo 1*

Http Request	Path	Tiempo respuesta (ms)	Throughtput
POST	/alta/iniciativa(ciudadano)	1186 ms	6,3 request/sec
POST	/alta/iniciativa(organizacion)	1155 ms	6,4 request/sec

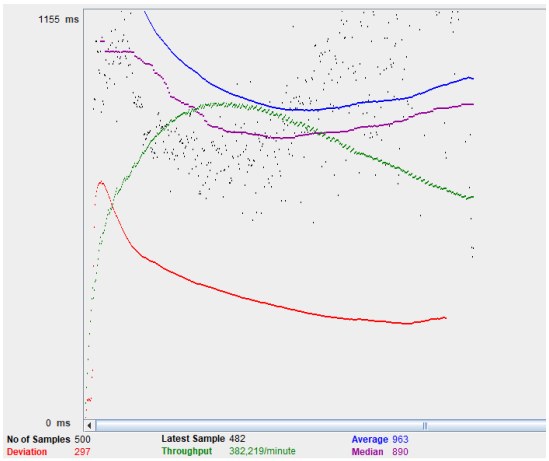


Figura 3: *Graph prueba 2 flujo 1(organización).*

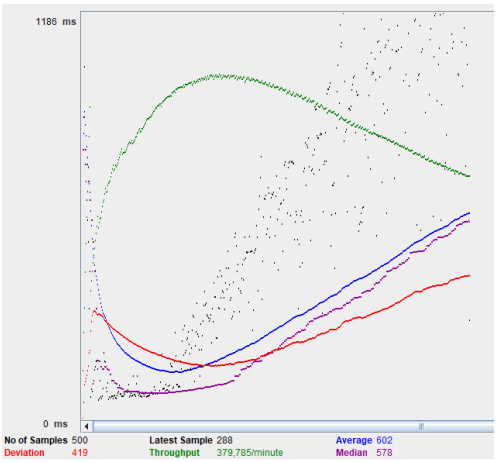


Figura 4: *Graph prueba 2 flujo 1(ciudadano).*

Para la siguiente prueba se utilizó al usuario ciudadano del sistema "colo@gmail" para crear 50 iniciativas en 10 usuarios. Se espera obtener al menos un certificado de creación para el usuario del sistema.

Tabla 5: Resultados obtenidos flujo 2

Http Request	Path	Tiempo respuesta (ms)	Throughtput
POST	/alta/iniciativa	1176 ms	10,5 request/sec
GET	/usuario/listarCertificados/{user}	1160 ms	10,7 request/sec

Label	# Samples	Error %	Throughput
get listar Certificados ...	500	1,80%	10,7/sec
TOTAL	500	1,80%	10,7/sec

Figura 6: *Tabla reporte prueba 2 flujo 2 (certificados).*

El 1,80 % de error representa los primeros 9(500*0.018) intentos que el usuario no tiene certificado, al igual que la figura 2. Además, el usuario obtuvo sus dos niveles de certificación como era esperado.

Tabla 6: Resultados obtenidos flujo 3

Http Request	Path	Tiempo respuesta (ms)	Throughtput
GET	we-Participation.web.elasticloud.uy	6 ms	195,1 request/sec
GET	/iniciativa/listar	23 ms	195,6 request/sec
GET	/proceso/listar	22 ms	195,8 request/sec

4.4. Prueba 3

- 50 usuarios - 100 iteraciones [**5000 muestras**]

Tabla 7: *Resultados obtenidos flujo 1*

Http Request	Path	Tiempo respuesta (ms)	Throughtput
POST	/alta/iniciativa(ciudadano)	17185 ms	3,0 request/sec
POST	/alta/iniciativa(organizacion)	11074 ms	3,0 request/sec

Label	# Samples	Error %	Throughput
post alta iniciativa user	5000	0,00%	3,0/sec
TOTAL	5000	0,00%	3,0/sec

Figura 7: *Tabla reporte prueba 3 flujo 1 (ciudadanos).*

Figura 8: *Count(*) iniciativas*

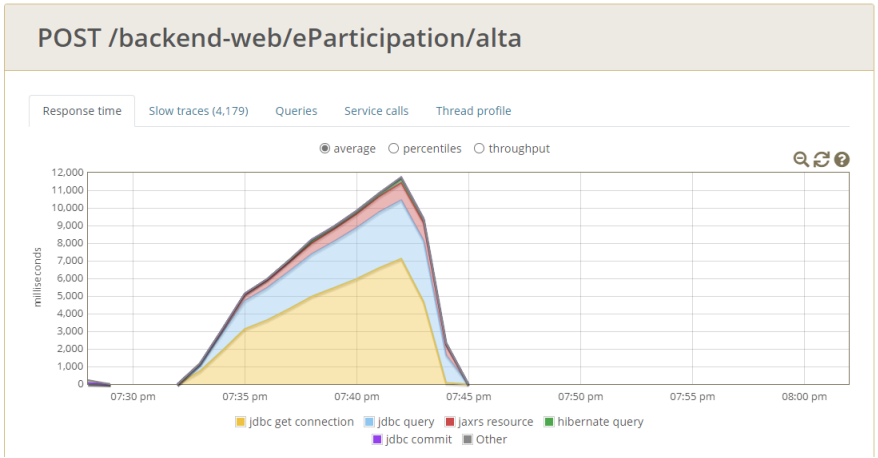


Figura 9: *APM prueba 3 flujo 3.*

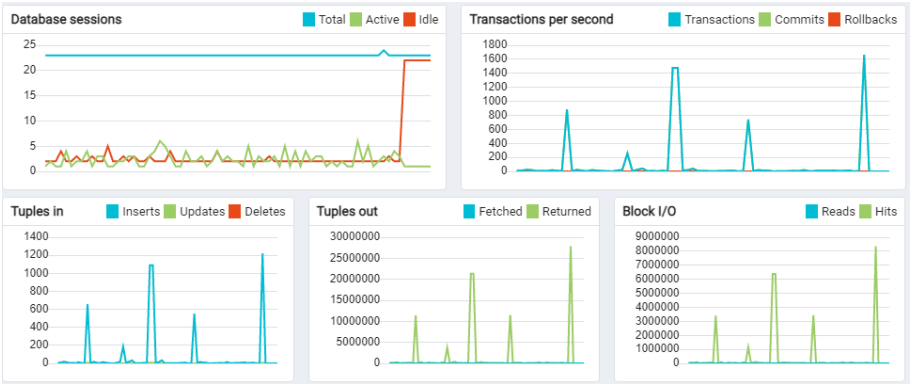


Figura 10: *Metricas PostrgreSQL al terminar prueba.*

- Se omite la prueba 3 flujo 2 debido a que el resultado está implicito en la prueba anterior.

Tabla 8: Resultados obtenidos flujo 3

Http Request	Path	Tiempo respuesta (ms)	Throughtput
GET	we-Participation.web.elasticloud.uy	19 ms	6,6 request/sec
GET	/iniciativa/listar	7090 ms	6,6 request/sec
GET	/proceso/listar	3083 ms	6,6 request/sec

Label	# Samples	Error %	Throughput
request dashboard	5000	0,00%	6,6/sec
TOTAL	5000	0,00%	6,6/sec

Label	# Samples	Error %	Throughput
get iniciativas	5000	0,00%	6,6/sec
TOTAL	5000	0,00%	6,6/sec

Label	# Samples	Error %	Throughput
get procesos	5000	0,00%	6,6/sec
TOTAL	5000	0,00%	6,6/sec

Figura 11: *Totales JMeter Prueba 3 flujo 3.*

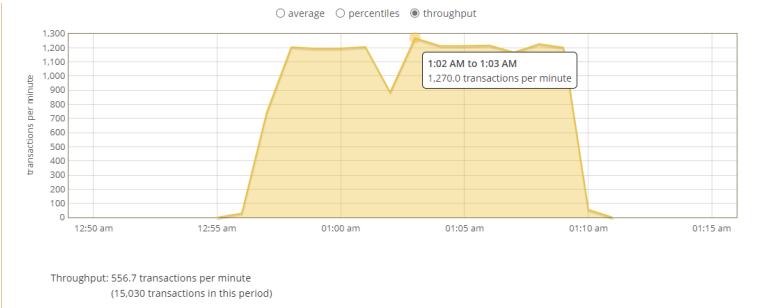


Figura 12: *Rendimiento APM Prueba 3 flujo 3.*

4.5. Punto de quiebre

Se buscó el punto de quiebre sin éxito de la aplicación incrementando los usuarios simulteánemente permitidos por JMeter, se logró hacer pedidos al dashboard de hasta 10000 usuarios, teniendo un tiempo de respuesta promedio de 20ms. Luego de los 10000 usuarios, el comportamiento de JMeter se volvió tedioso y con fallas decidiendo frenar la prueba de breakpoint.

Dado el inconveniente presentado por la herramienta JMeter, se toma la decisión de probar la solución en una máquina virtual, la cual se considera con menos recursos que el servidor actual. Se instala un clon del ambiente de pre-producción en un servidor virtual con las siguientes caracteristicas:

- Virtualizador: Oracle VM VirtualBox
- CPU: Intel Core i7-4930K @ 3.40GHz x4
- SO: Windows 11 64-bit
- RAM: 8 GB
- SSD: 127 GB

Al realizar nuevamente las pruebas de stress se detecta, probando incrementalmente la carga inicial de 1 usuario en 10 hilos porcentualmente hasta llegar al momento de tener activos 6384 pedidos simulteánemente el sistema deja de tomar solicitudes y empieza a rechazar debido a caída de sistema.

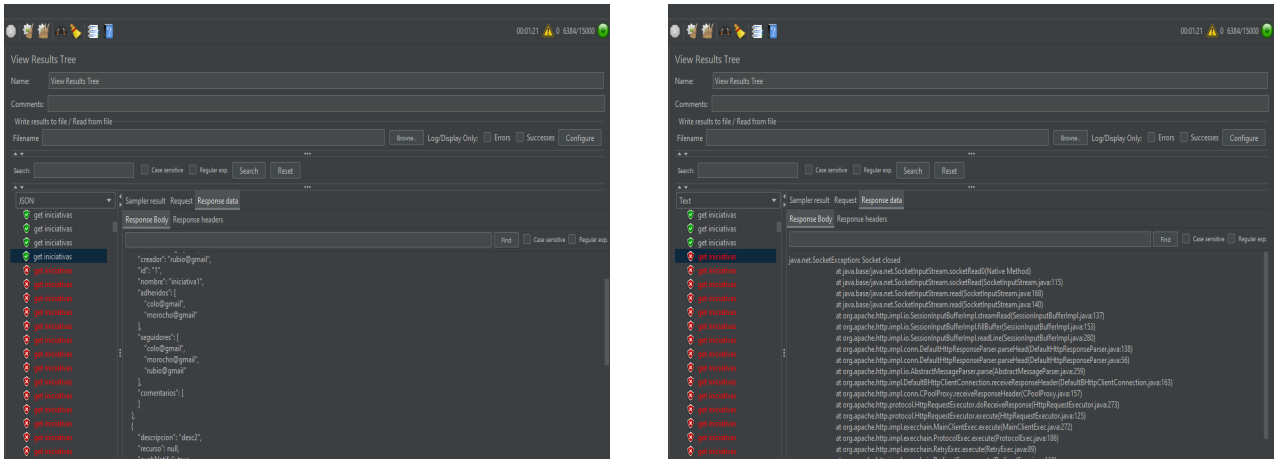


Figura 13: *Buscando breakpoint JMeter.*

4.6. Elastic cloud

Se verificó que la carga se distribuye por los nodos configurados al escalar horizontalmente. Cuando el servidor de aplicaciones detecta que tiene un uso de CPU, RAM o I/O elevado durante algunos minutos sin cesar, se crean automáticamente nodos auto-configurables para que el rendimiento de la aplicación no se degrade. Se aprecia como cuando la aplicación empieza su stand-by, los nodos sin usar se van quitando a medida de un uso con tendencia a la baja.

5. Optimizaciones

En base a los resultados obtenidos en las pruebas hechas se realizaron mejoras de distintos niveles con el objetivo de mejorar el rendimiento de eparticipation.uy.

5.1. Cuellos de botella

Se aprecia en la figura 9, que **"jdbc connection"(Bases de datos PostgreSQL)** ocasiona un cuello de botella en el sistema debido a la cantidad de transacciones que realizan el alta iniciativa. Se minimizaron las transacciones que llaman al contexto de persistencia y se mantiene el contexto transaccional JTA.

5.2. Concurrencia

Se detecto en la prueba 2; flujo 3, un uso concurrente de los pedidos de *READ* en el contexto transaccional. Se agrego @Lock(Lock.READ) para permitir el acceso simultaneos de pedidos que todavia no se hayan completado.

```
@Singleton
@Startup
@Lock(LockType.READ)
public class Persistencia implements PersistenciaLocal {

    @PersistenceContext(unitName = "eParticipationDB")
    private EntityManager em;
```

Figura 14: *Manejo de concurrencia Persistencia.java.*

5.3. Entorno JVM

Durante las pruebas de carga se detectó un uso elevado del Garbage Collector(GC), el cual era causado debido al tamaño maximo asignado por defecto de JBoss al heap de momoria JVM. Se cambiaron las propiedades de asignación de memoria en JVM de 512MB a 1024MB y se aprecia un ligero cambio en la utilización del GC. Luego se cambio nuevamente a 2048MB para que el servidor pueda fluir mejor y por consecuente llamar menos veces al GC.
Con el argumento **-Xmx2048m** se asigna a la JVM memoria máxima disponible.

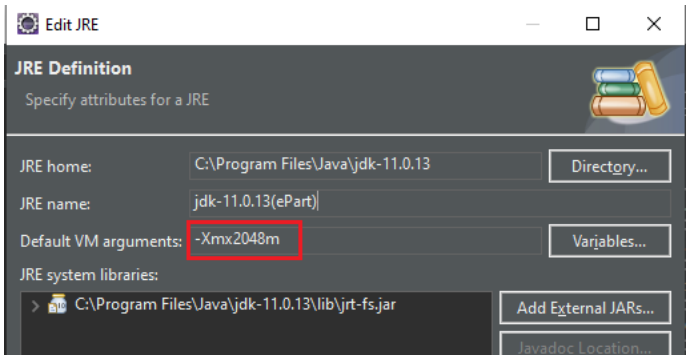


Figura 15: *Configuración eclipse JRE Enviroments.*

5.4. SQL

Se realizaron cambios en los procesos clave de la aplicación debido a que se encontraron usos que degradaban la perfomance de la aplicación. - Como por ejemplo no ejecutar la query de setear las relaciones de un ciudadano si sus listas eran = 0. Se detecta que las sesiones de jdbc(Figura10) son muchas, esto se debe al comportamiento por defecto de la anotación *@EJB*,

la cual maneja el ciclo de vida de los componentes y no se puede dejar una sesión inicializada del PersistenceContext incluso siendo *@Singleton*. Se probaron varias alternativas como usar el *type.PersistenceContextExtended*, pero tiene conflictos con el tipo de transaccion que usamos JTA. Además se intento crear una única instancia del EntityManager sin éxito debido a que luego de un commit la conexión se cierra por su lifecycle JTA. [5] [6]

5.5. MongoDB

Se detectó que la aplicación creaba muchas conexiones al mismo servidor de mongodb. Se cambió el constructor que instancia la conexión nosql con el cluster de Atlas para que el mismo sea singleton y use siempre la misma conexión que ya se inicializó por algún usuario. Se crearon índices para las colecciones Instrumento Y Comentarios, con el fin de buscar por autor y no por id generado automaticamente por mongodb; con este cambio se logra encontrar mas rápido los comentarios y respuestas a los distintos tipos de instrumento de un usuario X.

Referencias

[1] Kent Beck Erich Gamma. *Java unit tests*. Inf. téc. URL: https://junit.org/junit4/.

[2] Marc R. Hoffmann. *Java Code Coverage for Eclipse*. Inf. téc. URL: https://www.eclemma.org/.

[3] Apache. *Apache JMeter*. Inf. téc. URL: https://jmeter.apache.org/.

[4] OpenSource. *Glowroot APM*. Inf. téc. URL: https://glowroot.org/.

[5] *EJB Persistence Context*. Inf. téc. URL: https://www.infoworld.com/article/2072181/extended-persistence-context-in-stateful-session-beans.html.

[6] *Ciclo de vida JPA*. Inf. téc. URL: https://thorben-janssen.com/entity-lifecycle-model.