*María Camila Hernández*

*Daniela Mariño Rodríguez*

*Andrés Felipe Moreno Marín*

*Juan Manuel Lizarazo*

# Experimento 2 parte 2

# Se analizan y comparan los resultados de las pruebas de carga junto con los respectivos gráficos con un énfasis particular en los porcentajes de error.

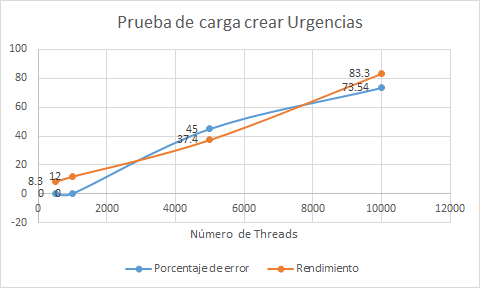
Nueva prueba de carga:











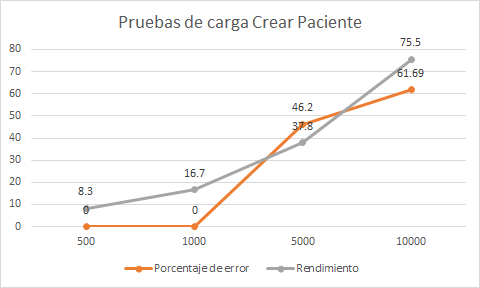
Se vio una ligera mejora en las pruebas de carga en esta instancia, ya que anteriormente había un inconveniente con el ID en urgencias, lo creaba repetidamente y no dejaba crearlo secuencialmente la base de datos.







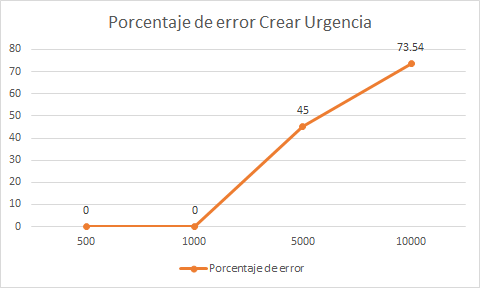
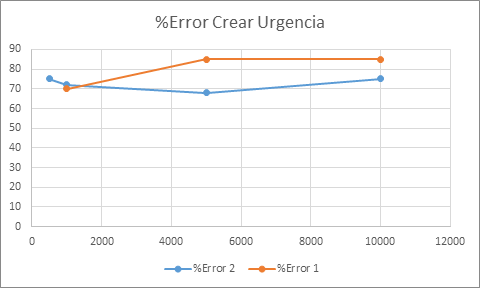


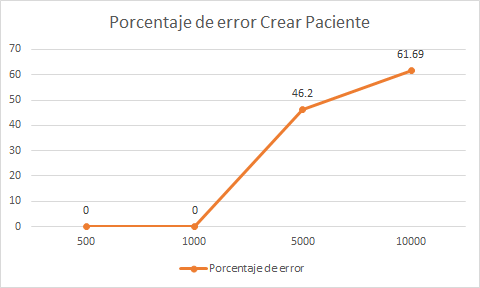
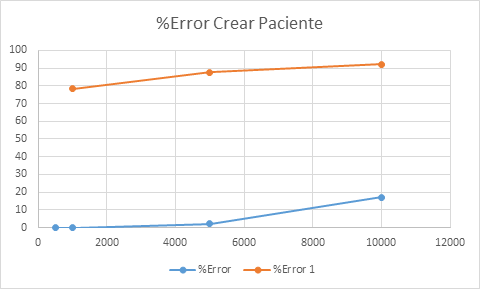


En este caso, las pruebas de carga presentaron una mejora frente a los resultados obtenidos anteriormente, mínimos problemas fueron corregidos y el balanceador de carga fuer implementado, para garantizar esta mejora en la arquitectura de la aplicación

# Se analizan y comparan los resultados de las pruebas de carga de la entrega anterior con los de la entrega actual

Los resultados de estas pruebas fueron mejores que en la del caso anterior, evidenciando una mejora sustancial en la arquitectura de la aplicación, la velocidad de respuesta y el tiempo que toma para realizar las actividades que le mandamos.



,

En la columna izquierda contamos con la gráfica que corresponde al porcentaje de error de la prueba pasada, y a su derecha contamos con la gráfica de esta entrega. Aunque en Pacientes se generó un leve déficit en este aspecto, se debe tener en cuenta que los errores en urgencias se debieron a un mal manejo del Id en la base de datos, lo que hizo el porcentaje de error mayor a lo que se debió obtener. Ahora, con las gráficas actuales, vemos una mejora substancial y una estandarización de resultados al ver los porcentajes de error muy cercanos el uno al otro, cumpliendo el objetivo de mantener la aplicación corriendo en diferentes servidores con un balanceador de carga y garantizando su funcionalidad, aunque una de las maquinas deje de correr.

Con una seguridad implementada, se puede esperar que los resultados de la pruebas de carga se vean afectados, ya que con la capa de seguridad, se espera que se tenga que autenticar el envió o recepción de datos, afectando el rendimiento en la medida en que ya no se van a procesar tantos datos en poco tiempo aunque el porcentaje de error no se vea afectado. Sin embargo, la aplicación va a ser mucho mas segura, cumpliendo estándares de calidad fundamentales y dejando ver que es una aplicación bien implementada.

# Decisiones de arquitectura

Con respecto a lo propuesto en la entrega 1 del SAD, los escenarios de confiabilidad y autenticacion deben ser capaces de garantizar que solo los medicos y el paciente puedan acceder a la historia clinica del paciente y que los usuarios se autentifiquen al menos una vez antes de acceder a los recursos. Para garantizar estos escenarios la aplicación cuenta con :

## Patrón arquitectónico

* 1. **Arquitectura orientada a eventos (EDA):** Esta arquitectura es de gran utilidad en sistemas con necesidades de un alto grado de reacción ante cualquier evento no ordinario, lo que es muy relevante para nuestro proyecto con respecto al tema de urgencias. Ya que cuando una urgencia es reportada en el sistema lo que se espera es poder responder según lo estipula el escenario de calidad de desempeño. Esto lo logramos mediante la vista controllers del proyecto ya que cuando se genera una urgencia, se transmite al controlador de la clase Urgencia quien se encarga de unir a las demás clases para responder lo más rápido posible a la necesidad del usuario. Adiconalmente, se prefirió la continuación con este patrón de diseño debido a que puede ser reforzado por medio de las diferentes decisiones de seguridad que se desean implementar al proyecto.

## Estilos arquitectónicos

* 1. **Balanceador de carga:** Favorece tácticas como la programación de recursos, múltiples copias de computación, múltiples copias de información. Lo anterior es pertinente para nuestro proyecto debido a que al contar con un balanceador de carga podremos reportar e informar a los servicios del hospital sobre la urgencia que está presentando cualquier paciente, ya que el balanceador se carga se encarga de asignar la petición al esclavo que se encuentre disponible para responder a la solicitud, por lo tanto, no es necesario esperar a que un solo servidor esté listo para responder.
  2. **Seguridad con Play-Authenticate:** Agrega en nivel de seguridad a los datos codificando o protegiendo toda la informacion que entra o sale de la aplicación con una estructura usuario contraseña, la cual permite identificar quien está haciendo la transaccion y si esta autorizado a hacerla. Esta implementacion tambien permite asegurar la informacion, para que sea protegida de cualquiera que trate de acceder a ella y no este autoizado a hacerlo, protegiendonos de personas o entidades que intenten añadir informacion falsa u obtener informacon delicada para el Hospital.

## Tácticas

Con los patrones y estilos arquitectónicos expuesto anteriormente se puede afirmar que la aplicación posee las siguientes tácticas que garantizan la seguridad:

* + **hashing:** Los servidores se comunican con la base de datos con md5, un algoritmo de reduccion criptografica que codifica los datos y grantiza que no se hayan modificado en el proceso.
  + **Autenticacion:** Al usar play-authenticate podemos garantizar que los usuarios de autentifiquen con su correo y contraseña para evitar roblemas de seguridad como spoofing o information disclosure.
  + **Autorizacion:** Al usar deadbolt podemos generar roles y darles permisos a esos roles, esto garantiza que no tengamos problemas de seguridad como elevation of privilege.

# Se evidencia un proceso basado en análisis de los requerimientos, diseño de elementos de arquitectura, diseño de la solución, implementación y pruebas. Se cuenta con artefactos que soportan el proceso y con un depósito de las fuentes.

**Implementación y pruebas, Diseño de la solución:**

**Artefactos:** El artefacto construido fue el descrito en el documento anterior, un sensor que guarda mediciones de diferentes tipos y las envía a la aplicación directamente, esto se hace guardando la información del sensor en variables, que posteriormente se convierten en un objeto JSON para poderlo enviar a la aplicación por la URL solicitada con el método PUT.

**Diseño de elementos de arquitectura:**

La base de datos en PostgreSQL fue implementada debido a las características que benefician nuestro proyecto, entre las que se encuentran la seguridad del acceso a la información y la forma de enviar la información. Se puede decir que estas características ayudaron a nuestro proyecto ya que garantizan que no se sufra de problemas de seguridad como information disclosure o tampering. Adicional a esto, en esta versión del proyecto se implemento un esquema de seguridad en los datos, para garantizar que la información que se envía a través de internet sea segura y no pueda ser accedida fácilmente por cualquier persona o entidad ajena a la institución. Esta arquitectura se llama "Play-Authenticate", la cual es muy adaptable a cualquier aplicación (escrita sobre la arquitectura play) a utilizar y cuenta con una generación dinámica de URLs para los servicios que se van a implementar. Los beneficios de esta implementación radican en que la unión de la información en la base de datos siempre se hace con autorización del usuario o automáticamente, para evitar algún problema con lo que se está agregando, el uso de una verificación con correo y contraseña para verificar quien está subiendo datos y si está autorizado a hacerlo, funciona completamente con HTTPS, garantizando la seguridad de la URL a manejar y la posibilidad de utilizar cuentas en diferentes dispositivos, para evitar que exista algún problema con la modificación de datos si no se encuentra frente a un solo ordenador o aplicación.