*María Camila Hernández*

*Daniela Mariño Rodríguez*

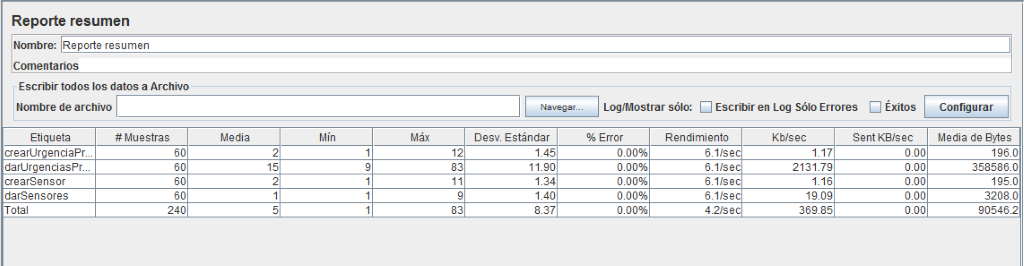
*Andrés Felipe Moreno Marín*

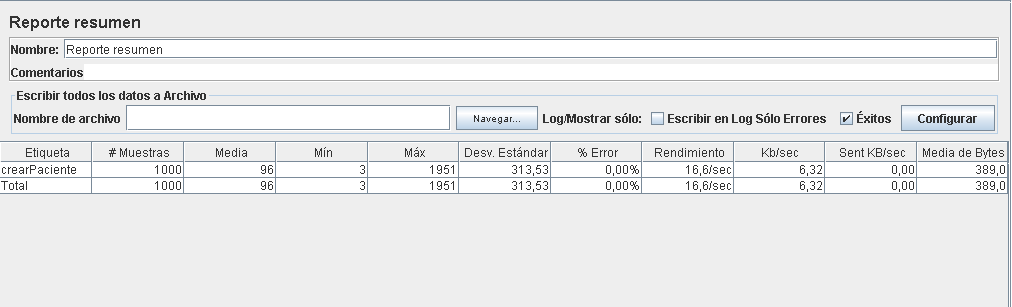
*Juan Manuel Lizarazo*

**Experimento 1 parte final**

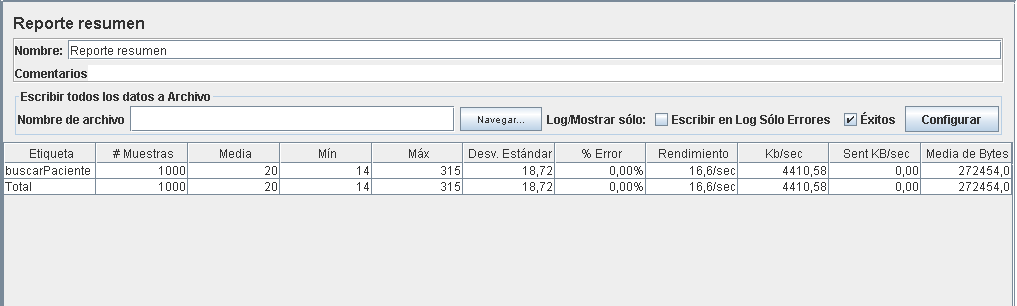
**Post-experimentación** **Experimento 1 Parte 1**

***Resultados obtenidos***

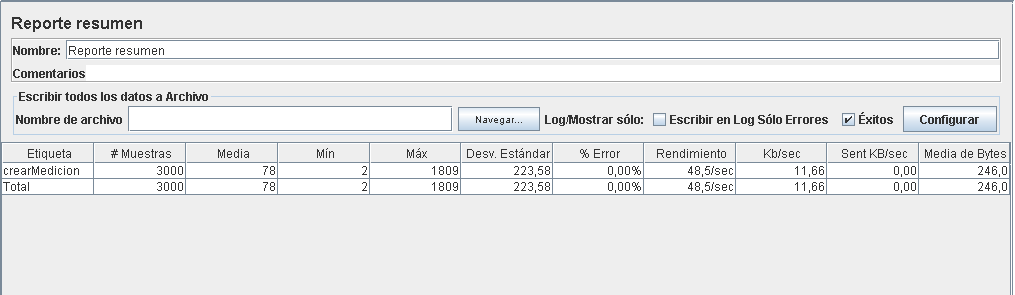
*Figura 1. Pruebas de carga Clase Urgencia y Sensor (Crear y Dar)*



*Figura 2. Pruebas de carga crear Paciente*



*Figura 3. Pruebas de carga buscar información Paciente*



*Figura 4. Pruebas de carga crear medición*

***Pruebas de Crear Urgencias en orden: 1000, 5000, 10000 y 50000***





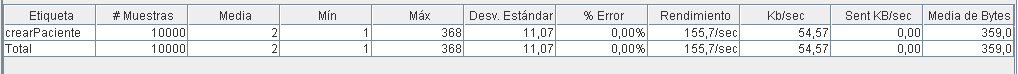




***Pruebas de Crear Paciente en orden: 1000, 5000, 10000 y 50000***







**Post-experimentación** **Experimento 1 Parte 2**

***Resultados obtenidos***

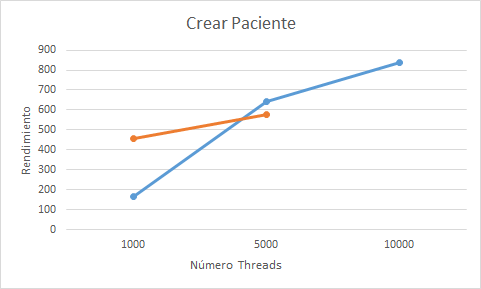
***Pruebas de Crear Paciente en orden: 1000, 5000, 10000 y 50000***

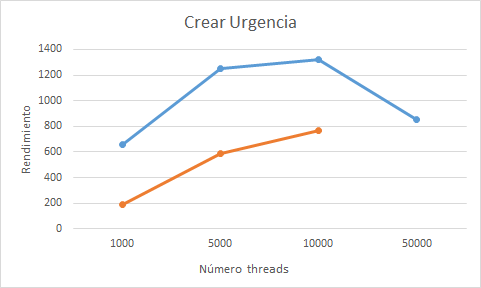


***Pruebas de Crear Urgencias en orden: 1000, 5000, 10000 y 50000***



***Resultados obtenidos entrega final vs Resultados sin persistencia***





Donde la línea azul representa como se comportaba la aplicación con el uso de mocks, mientras la línea naranja representa el comportamiento actualmente de la aplicación con persistencia.

***Artefactos construidos:***

Se construyeron los artefactos más sensibles para el diseño de la solución. Es decir, a partir de los requerimientos se puede determinar cuales de ellos realmente son críticos, como lo es una urgencia o el envió de los datos por parte de los sensores hasta la aplicación ya que a partir de los datos obtenidos por medio del dispositivo sabremos cual es la ruta que se debe seguir para atender al paciente. Es decir, dependiendo de la información se tienen unos patrones (colores) que determinan las acciones siguientes a la toma datos.

***Decisiones de arquitectura:***

Es necesario tener sumo cuidado con la información que es arrojada por los sensores ya que la vida de los pacientes puede estar en juego a causa de esto, por está razón con la experimentación surgen luces de hacia donde deberíamos ver con respecto a tecnologías y arquitecturas. Por ejemplo, después de está primera experimentación sabemos que es necesario un sistema asíncrono que funcione con una arquitectura de actores como es el caso de Play. Por eso la implementación del proyecto sobre este Framework. Además, se puede afirmar que los tiempos obtenidos son consistentes con las relaciones esperadas.

Por otra parte, en el tema de la persistencia decidimos implementar un base de datos relacional llamada PostgreSQL la cual tiene diversas características pertinentes para nuestro proyecto, entre las que se encuentran:

* ***Alta concurrencia:*** Esto permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Lo cual para nuestro proyecto es muy pertinente ya que si un doctor necesita consultar el historial de un paciente que acaba de entrar a urgencias y otros doctores están en consulta simultáneamente no se presenten problemas con la veracidad de la información ni cruces de información. Esta característica garantiza una visión consistente de lo último a lo que se le hizo commit.
* **Amplia variedad de tipos nativos:** Debido a que los tratamientos o consejos dados por parte del medico al paciente tienen una longitud muy variable, al tener texto ilimitado podemos garantizar que no habrán complicaciones a la hora de prestar el mejor servicio posible al usuario. Además, con la información guardada por parte de los sensores, es necesario tener una extensión considerable para cada paciente ya que las mediciones que se realizan pueden ser 2 o más veces al día con respecto a las variables de interés para el hospital como los son el de frecuencia cardiaca, presión sanguínea, nivel de estrés.
* **Disparadores (triggers):** Son objetos que se asocian con tablas y se almacenan en la base de datos. Estos se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado, ayudando a la administración de la base de datos. Son principalmente usadas para implementar las reglas de negocio (tipo especial de integridad) de una base de datos. Teniendo en cuenta que una restricción de negocio es cualquier restricción, requerimiento o necesidad especial que debe ser verificada al momento de intentar agregar, borrar o actualizar la información en una base de datos. Podemos afirmar que esta es una característica útil para el proyecto ya que nos dará la certeza de que cuando un medico por ejemplo deba modificar la información asociada a un paciente cumple con criterios específicos como garantizar su verificación como medico garantizando que efectivamente la información que será modificada en el hospital del paciente sea verídica y consistente con la situación real del paciente.

Se planea en un futuro implementar un balanceador de carga least-conn, el cual permitiría distribuir la carga entrante entre varios servidores. Esto haría referencia a una arquitectura Push y al ser least-conn los servidores no se atragantarían de peticiones, sino que estas serían distribuidas a los servidores que menos peticiones tiene.

**BIBLIOGRAFÍA**

<https://es.wikipedia.org/wiki/Trigger_(base_de_datos>)

<https://es.wikipedia.org/wiki/PostgreSQL#Ventajas>