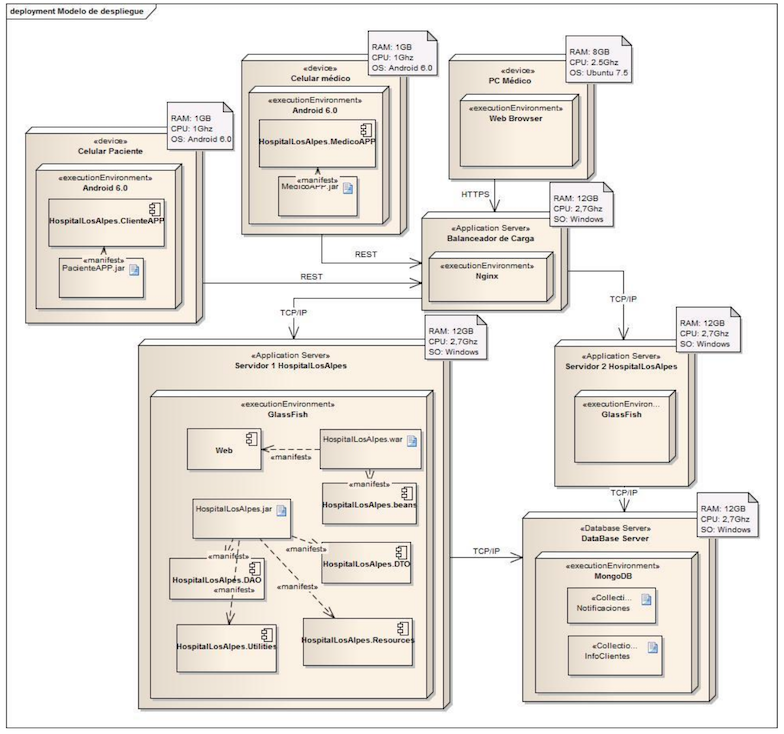
# Experimento 2 entrega 2

Con el fin de mejorar la disponibilidad del sistema, se optó por implementar una arquitectura como la del siguiente diagrama:



**Figura 1.** Arquitectura implementada

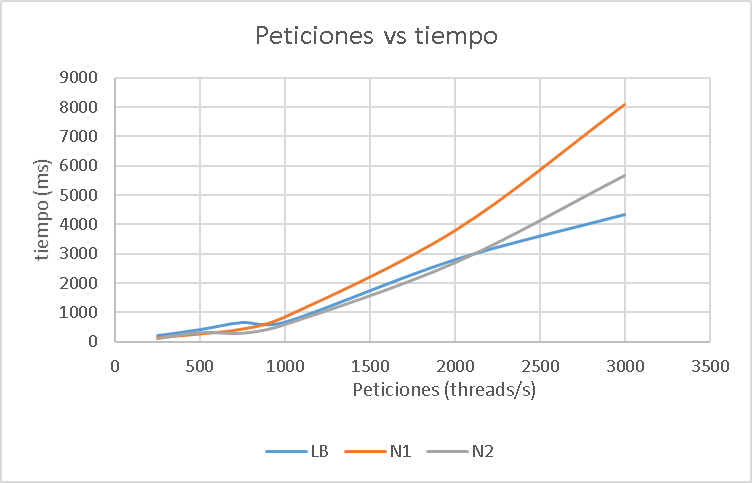
Se usaron 4 máquinas diferentes que ejecutaban peticiones sobre el balanceador de carga.

El balanceador de carga se configuró usando el software de nginx e implementando la técnica de leastConn, la cual elige el servidor con menos carga en el momento para asignarle la tarea. Adicionalmente se usó la página de error por default del balanceador en caso que ninguno de los servidores de aplicaciones respondiera. El servidor de aplicaciones 1 tenía la aplicación corriendo sobre el IDE de Eclipse, mientras el nodo 2 tenía la aplicación desplegada en Netbeans.

Las pruebas se hicieron simulando peticiones POST de notificación sobre la plataforma. Inicialmente, se apuntó al balanceador de carga y luego a cada uno de los nodos directamente, obteniendo los siguientes resultados:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Tiempo de respuesta (ms)** | | |
|  | **LB** | **N1** | **N2** |
| **250** | 217 | 147 | 107 |
| **500** | 421 | 273 | 313 |
| **750** | 667 | 441 | 294 |
| **1000** | 672 | 853 | 586 |
| **2000** | 2794 | 3791 | 2691 |
| **3000** | 4330 | 8090 | 5659 |

**Tabla 1.** Tiempos de respuesta método POST



**Figura 2.** Gráfica de los resultados de tiempos

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Porcentaje de error (%)** | | |
|  | **LB** | **N1** | **N2** |
| **250** | 0% | 0% | 0% |
| **500** | 0% | 0% | 0% |
| **750** | 0% | 0% | 0% |
| **1000** | 0% | 0% | 0% |
| **2000** | 4,35% | 7,93% | 0% |
| **3000** | 8,33% | 14% | 0% |

**Tabla 2.** Porcentajes de error

Como se puede observar en la Figura 2, los tiempos obtenidos con el balanceador son similares a los tiempos obtenidos apuntándole directamente al nodo 2, pero a partir de las 2200 peticiones por segundo, se observa que el balanceador empieza a obtener mejores tiempos de respuesta.

Adicionalmente se observa tanto en la Tabla 1 de tiempos como en la Tabla 2 de porcentajes de error, que el nodo 1 de la solución no tiene el mismo desempeño que el nodo 2 ya que este llega incluso a presentar errores cuando se envían 2000 peticiones, cosa que no sucede con el nodo 1. Sin embargo, al implementar el balanceador, el porcentaje de error asociado al nodo1 se disminuye casi a la mitad.

Con respecto a la entrega anterior, se tenían los siguientes tiempos de respuesta y porcentajes de error:

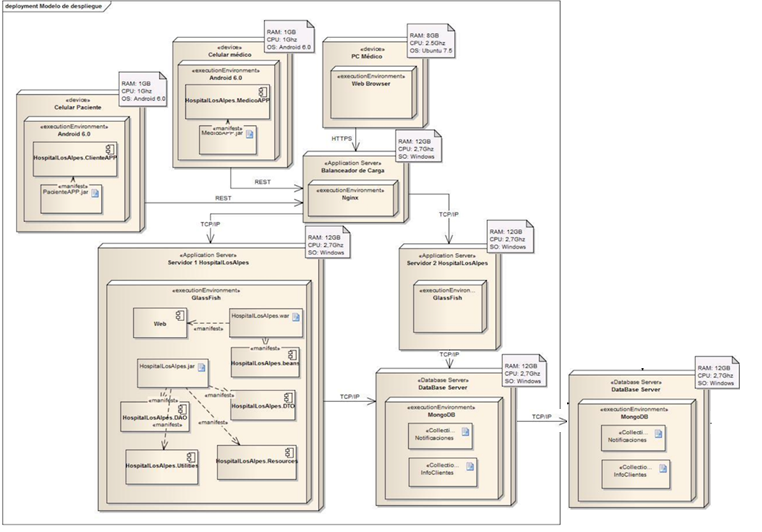
|  |  |  |
| --- | --- | --- |
|  | **Tiempo de respuesta (ms)** | **Porcentaje de error (%)** |
| **250** | 146 | 0% |
| **500** | 317 | 0% |
| **750** | 378 | 0% |
| **1000** | 735 | 0% |
| **2000** | 3248 | 0% |
| **3000** | 6886 | 0% |

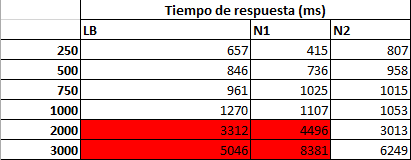
**Tabla 3.** Valores experimento 1

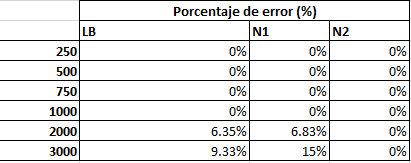
Se observa que los valores son similares a los obtenidos apuntando únicamente a un nodo.

Arquitectura nueva:

En cuanto a la arquitectura, se le aplico otro nodo de base de datos Mongo DB que simplemente replica lo que se guarda en el nodo principal, lo anterior para garantizar la integridad de los datos. ( el proceso de replicación no afecta al programa base) y para acceder a dichas bases de datos es necesario realizar una autenticación previa.







Lo anterior evidencia una clara subida de los tiempos de respuestas, así mismo una clara subida en el porcentaje de error. Esto se explica debido a que es necesario realizar una autenticación y se consume tiempo en este proceso.