**Entrega 2**

**Arquitectura y Diseño de Software**

**Candax**

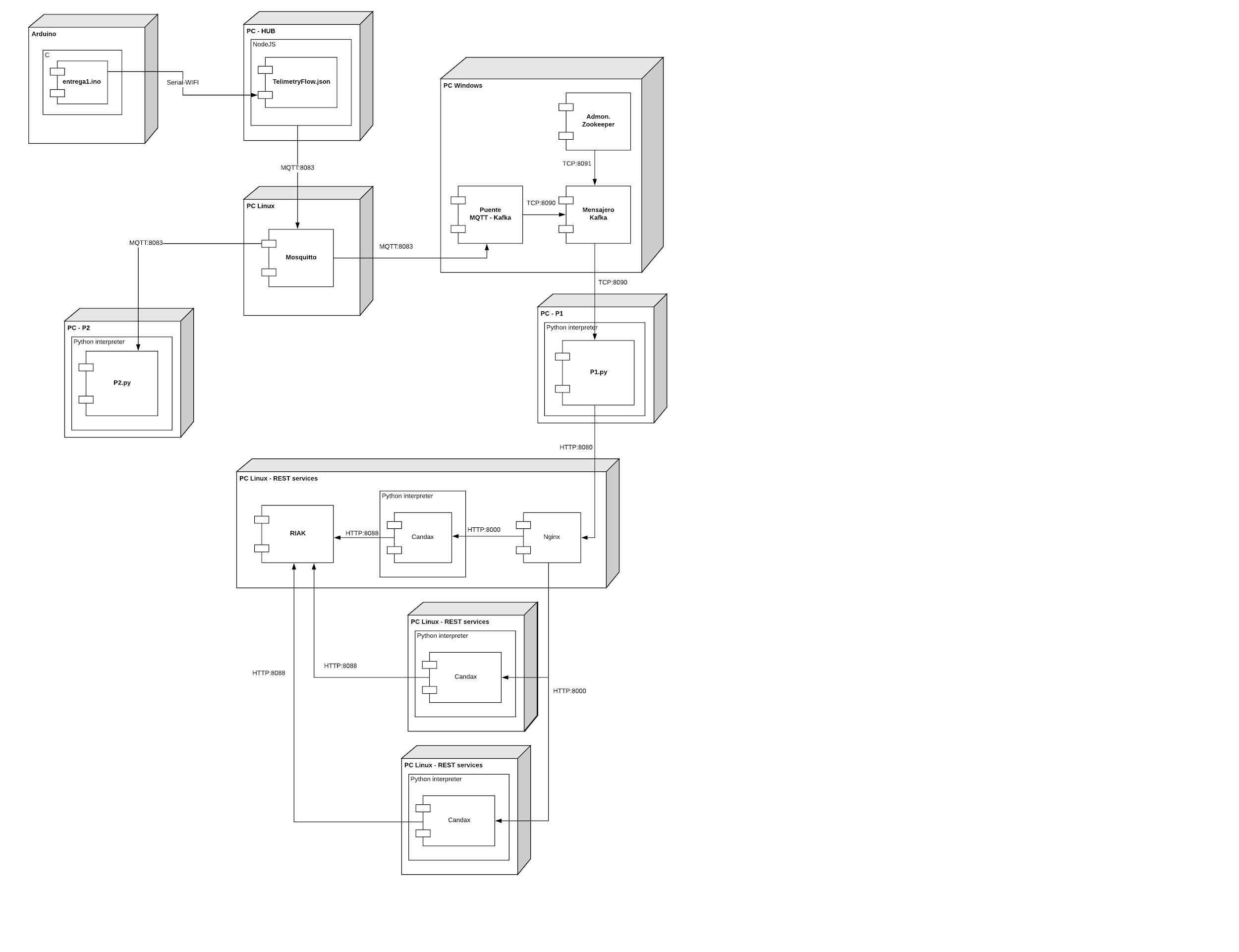
**Escenarios de prueba:**

Para persistir los datos se hace uso de Riak, una base de datos no relacional distribuida, que permite asegurar una alta disponibilidad y es escalable con facilidad. Con ayuda de Postman y JMeter se realiza carga de datos que permiten poblar Riak. Para cada una de las clases definidas en el modelo UML, que se encuentra dentro de la documentación, se crea una tabla de tal forma que se faciliten las consultas que se pueden realizar según el modelo de negocio. Para Propietarios, Administradores, Unidades Residenciales, Casas (Inmuebles), Hubs y Cerraduras se cargan 100 elementos con el fin de poder realizar las pruebas de carga para la petición de consulta.

Por otro lado, se realizan pruebas en Postman que permiten evaluar y comprobar la funcionalidad de las distintas acciones. Se prueban los siguiente escenarios en Postman:

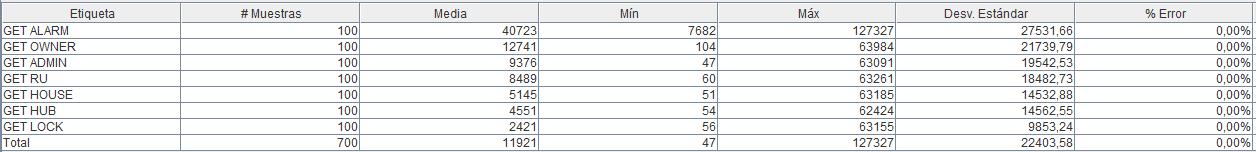
* CRUD Unidad residencial
* CRUD Inmueble (Casas)
* CRUD Hubs
* CRUD Cerraduras
* Creación y consulta de alarmas para YALE
* Creación y consulta de alarmas para un Propietario
* Creación y consulta de alarmas para un administrador

**Diagrama de despliegue:**

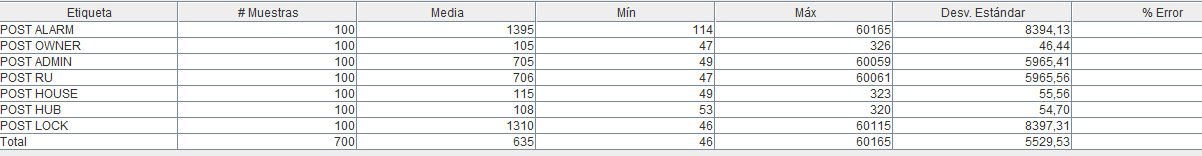


**Pruebas**

* **GET**

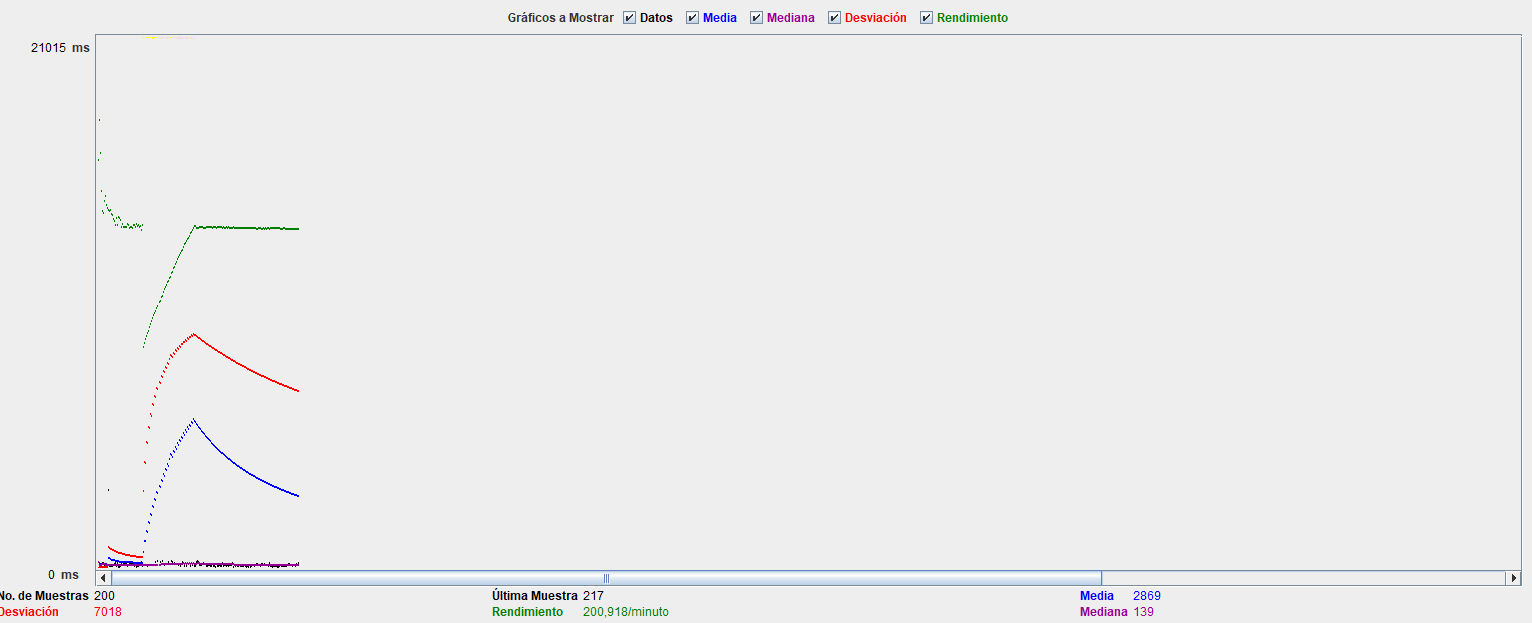
****

* **POST**

****

**Comparación de los resultados esperados versus los obtenidos con respecto a tiempo de respuesta y errores para 200 y 300000:**

* Resultados para 200





* Errores para 300000

Para esta prueba es necesario realizar un cluster para correr remoto JMeter. Adicionalmente, en el nodo maestro se asigna un mayor número de puertos con el comando *netsh int ipv6 set dynamicport udp start = 10000 num = 30000.* Las pruebas se realizan con 1, 2 y 3 nodos activos a los cuales el balanceador de carga redirige las solicitudes. Se realizan pruebas POST para alarmas para verificar que se cumplan con el escenario de calidad planteado.

|  |  |  |
| --- | --- | --- |
| Número nodos | Media (milisegundos) | Error |
| 1 | 77896 | 76% |
| 2 | 59007 | 54% |
| 3 | 39754 | 28% |

**Reflexión sobre si el diseño actual satisface los escenarios de calidad y en caso que no cómo planean modificar la arquitectura:** el diseño establecido no satisface los escenarios de calidad planteados. Se hace necesario realizar una mejor implementación del balanceador de carga e implementar buenas prácticas tanto en en la ejecución de las peticiones con el intérprete de Python, como en el manejo de los datos con Riak.

Algunos parámetros que se pueden manejar usando riak, es aumentar el número de nodos para poder replicar la información de la base de datos y que haya una alta disponibilidad y una alta tolerancia a fallos. Esto se debe, a que en el caso donde uno de los nodos no funcione de forma correcta o esté ocupado con una solicitud, otro puede funcionar. De igual forma, se quiere conocer si al tener el servicio distribuido entre la misma red hay una mejoría en cuanto al desempeño de la aplicación. Sin embargo se espera que la escalabilidad horizontal en la base de datos aumente el desempeño lo suficiente para cumplir con el requerimiento.

**Declaración del esfuerzo de cada miembro en exp. 1 intermedio y final:** el archivo se encuentra en el repositorio del grupo.