Proyecto de Curso

Entrega Final

Introducción a los Sistemas Distribuidos

Ingeniería de Sistemas

Estudiante:

Juan Sebastian Galeano Gonzalez

Docente:

Mariela Josefina Curiel Huérfano

3 de abril de 2024

Índice

1. Introducción.

Introducción

Objetivos del Proyecto

El presente proyecto tiene como objetivo principal el desarrollo de una solución a un problema utilizando una estructura distribuida, incorporando los conceptos de Cloud, Fog y Edge Computing.

Objetivos específicos.

- Desarrollar una solución a un problema de estructura distribuida.
- Utilizar patrones de comunicación síncronos y asíncronos.
- Resolver problemas presentes en sistemas distribuidos, tales como fallas en los componentes y persistencia de datos.
- Reconocer atributos de calidad (e.g., desempeño, resiliencia) asociados a la implementación de un sistema distribuido.

Contexto:

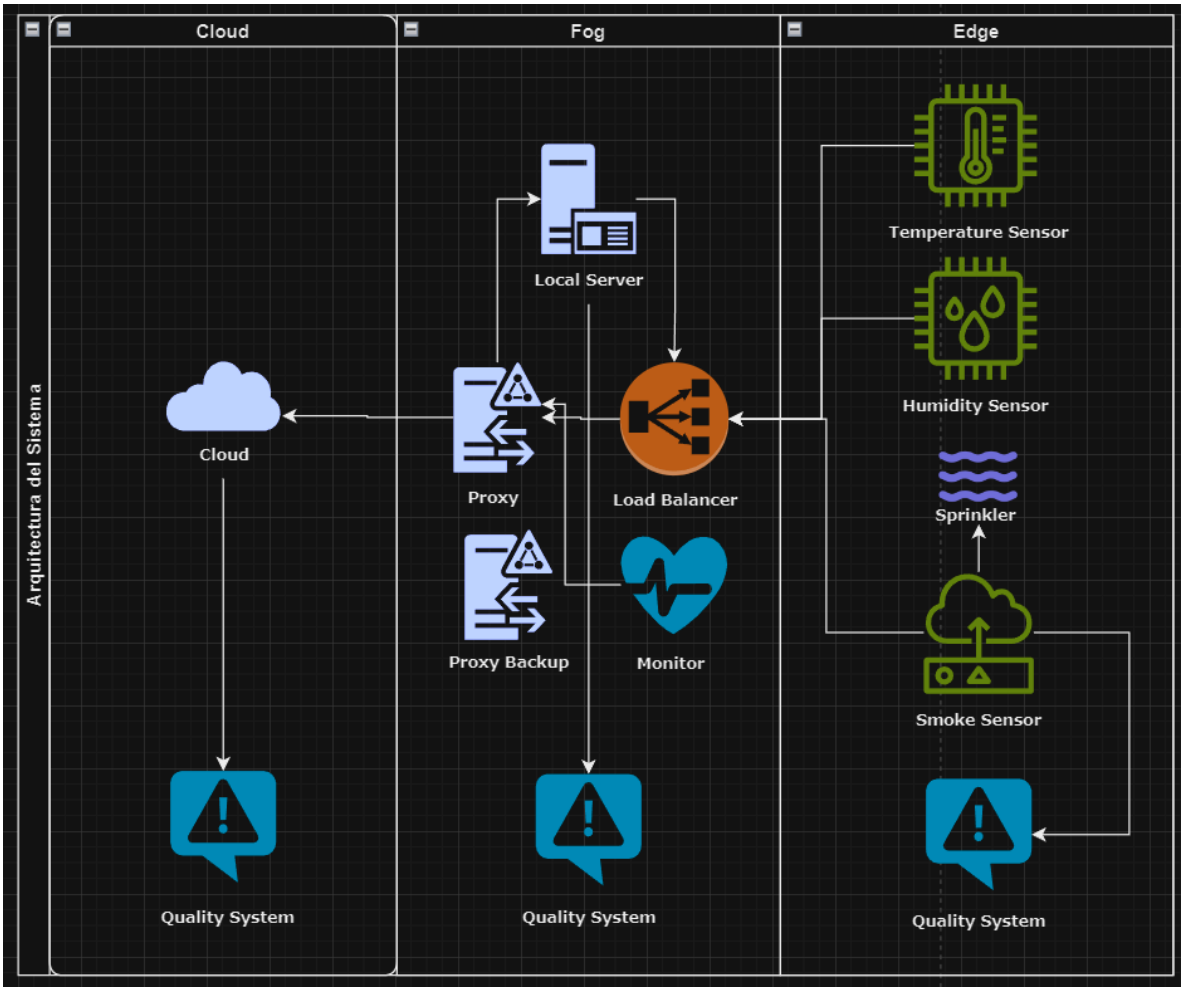
En las últimas décadas, los incendios forestales han aumentado considerablemente debido al cambio climático y a malas prácticas, tanto intencionales como no intencionales, por parte de algunos ciudadanos. Estos incendios tienen efectos devastadores: destruyen la vida silvestre y la vegetación, deterioran los suelos que pueden tardar siglos en recuperarse, agotan las fuentes de agua y aumentan la contaminación ambiental.

Por lo tanto, es crucial monitorear continuamente el estado de los suelos en las zonas de bosque para detectar aumentos de temperatura y cambios en las condiciones del ambiente. Esto permitiría controlar a tiempo un posible incendio. En este proyecto se propone realizar un sistema distribuido utilizando los conceptos de Edge, Fog y Cloud Computing para simular la tolerancia a fallas y la captura y análisis de tres parámetros críticos en una zona boscosa de los cerros orientales de Bogotá:

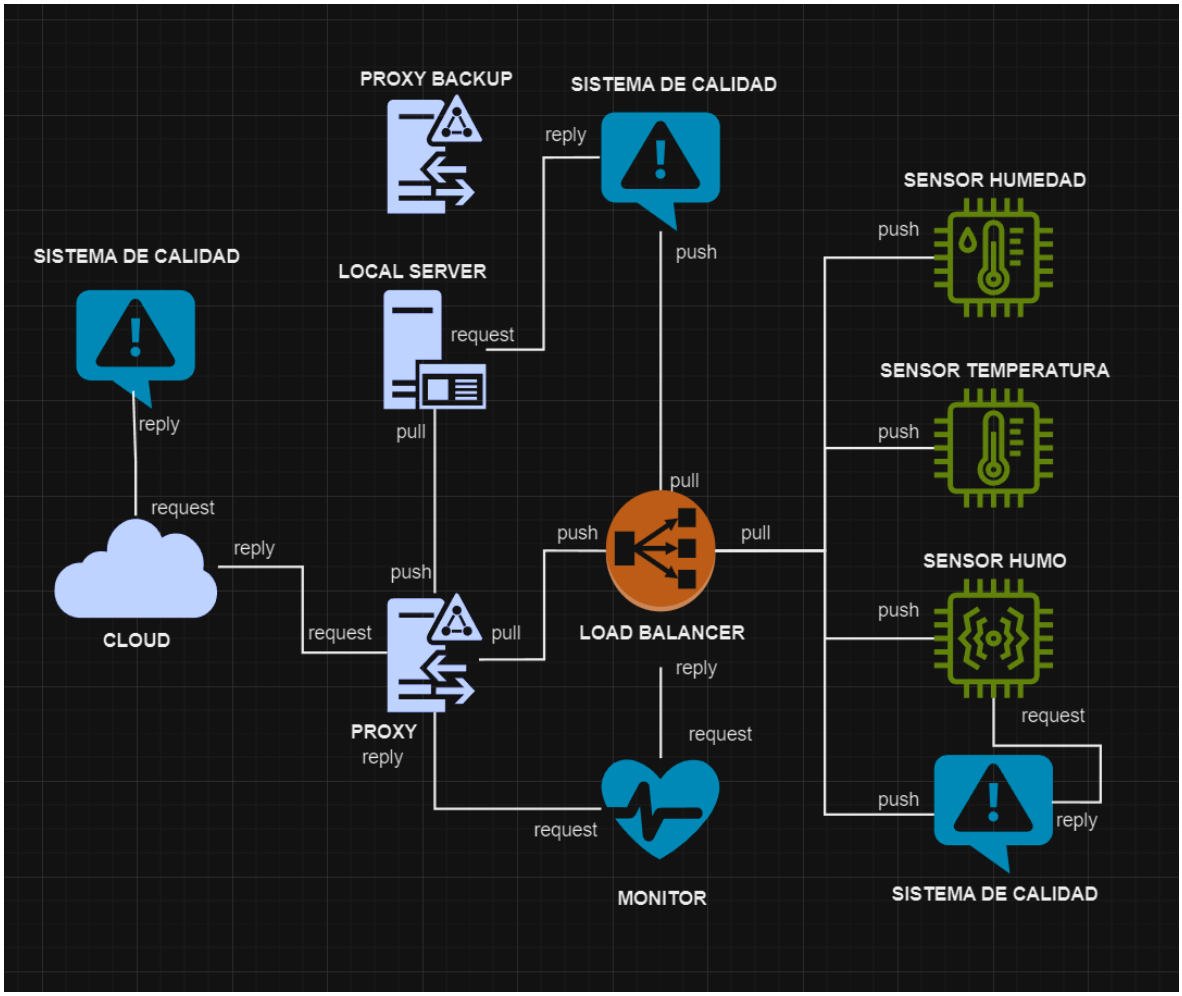
- Temperatura del aire: Rango entre 11°C y 29,4°C.
- Presencia de humo: Valor booleano (Verdadero/Falso).
- Humedad relativa: Promedio de 70%, con valores inferiores indicando posibles anomalías.

La solución propuesta involucra la implementación de sensores en la capa Edge, un proxy en la capa Fog y almacenamiento y análisis de datos en la capa Cloud. Estos componentes trabajan conjuntamente para detectar condiciones que podrían desencadenar un incendio forestal y activar mecanismos de respuesta inmediatos, como aspersores, además de almacenar datos para análisis posteriores y generación de estadísticas.

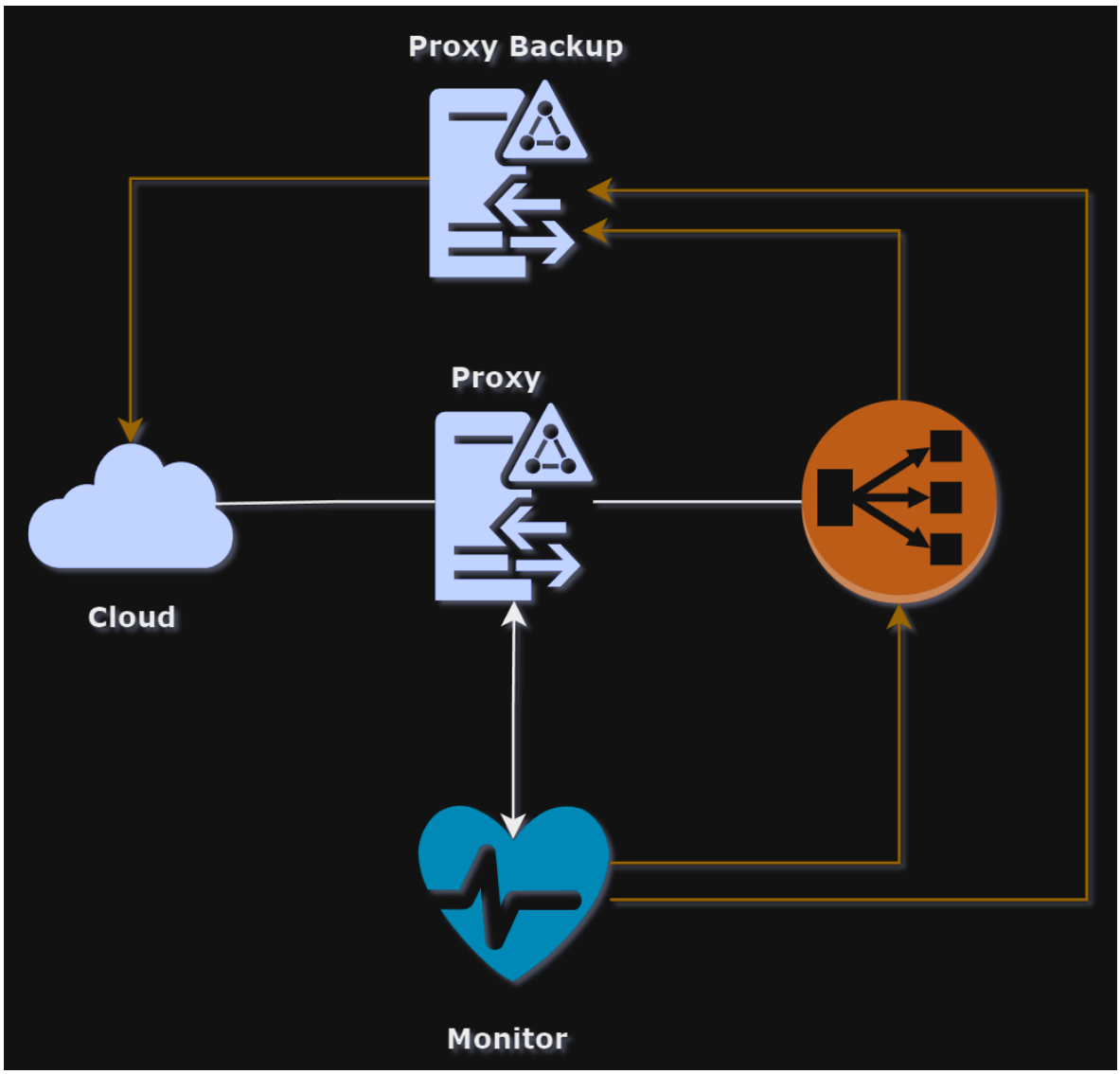
2. Modelos del Sistema.
    a. Arquitectura del Sistema:



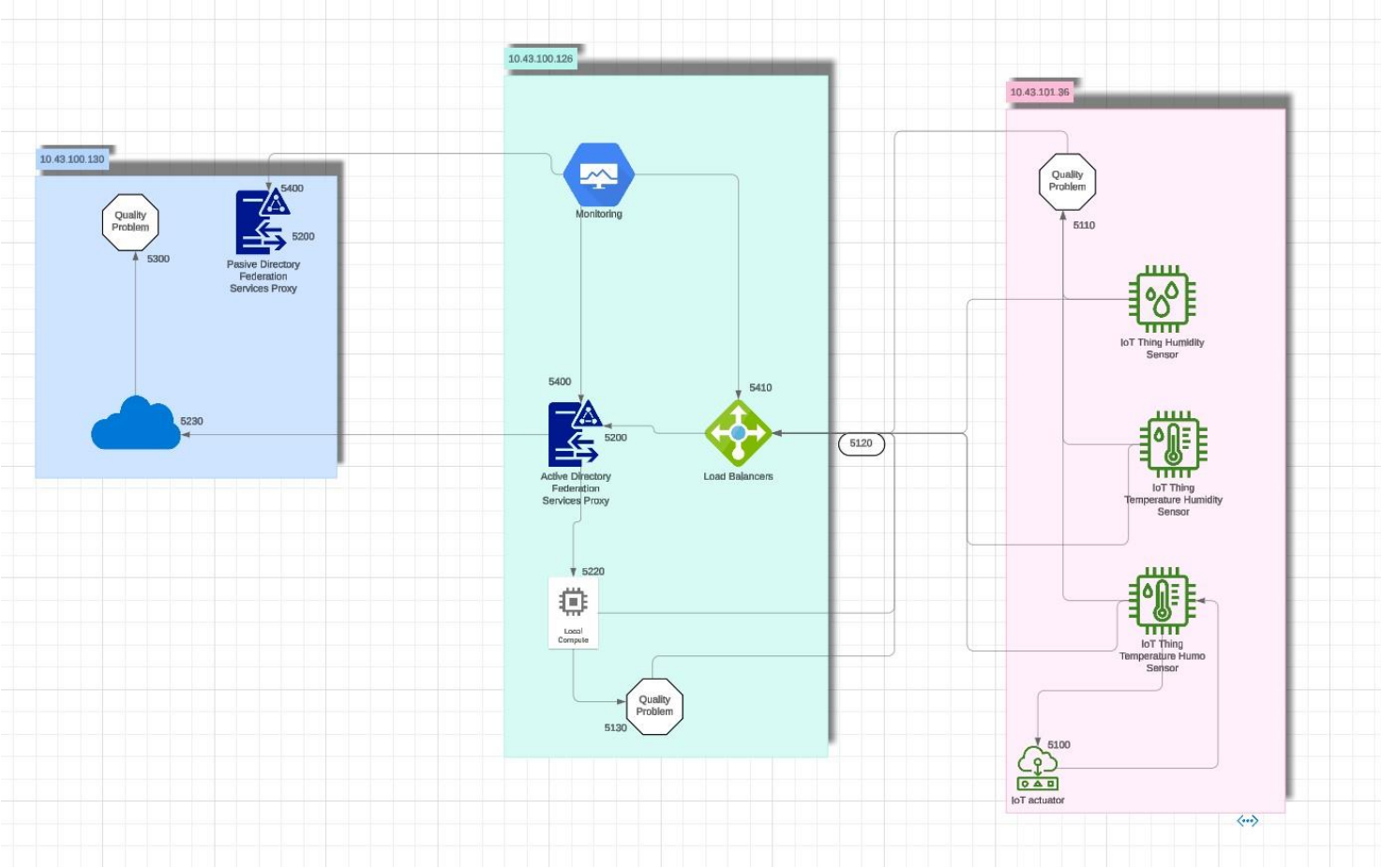
b. Diagrama de Interacción:



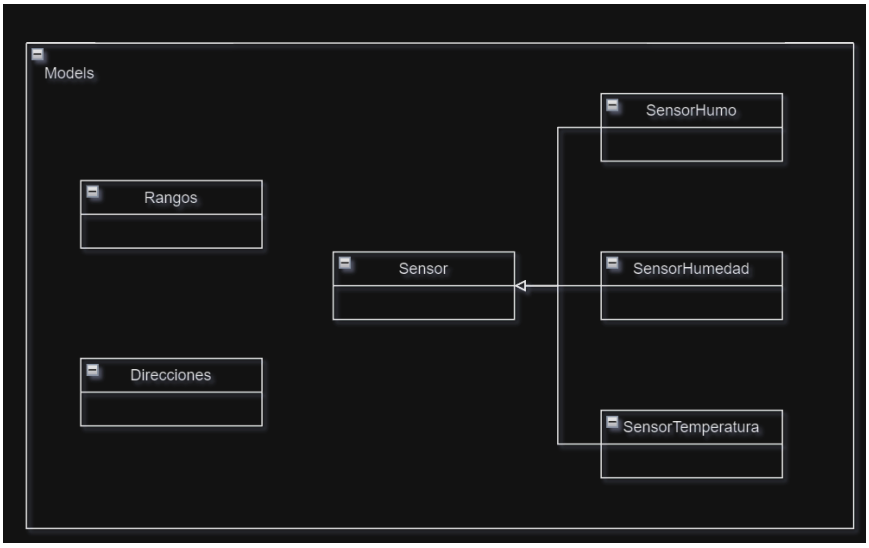
c. Diagrama de Fallos:



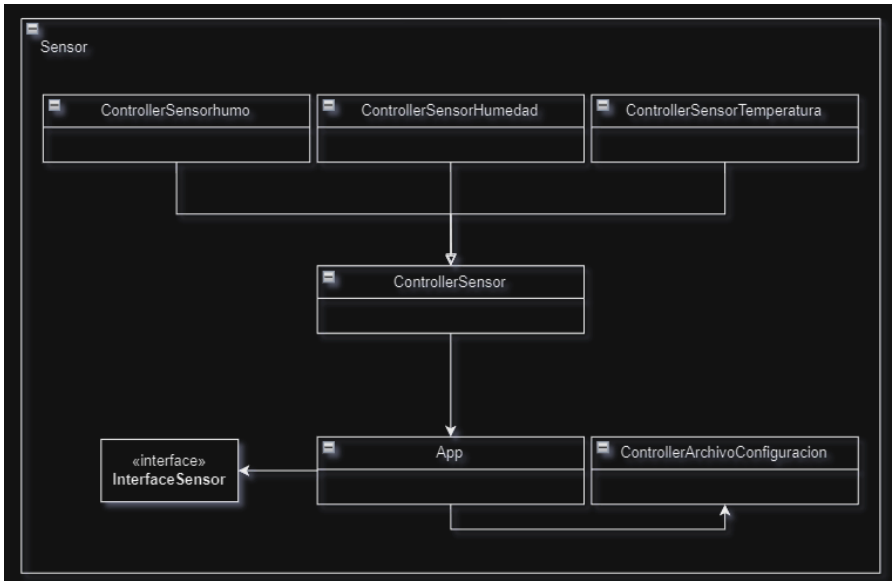
3. Diseño del Sistema.
    a. Diagrama de Despliegue:



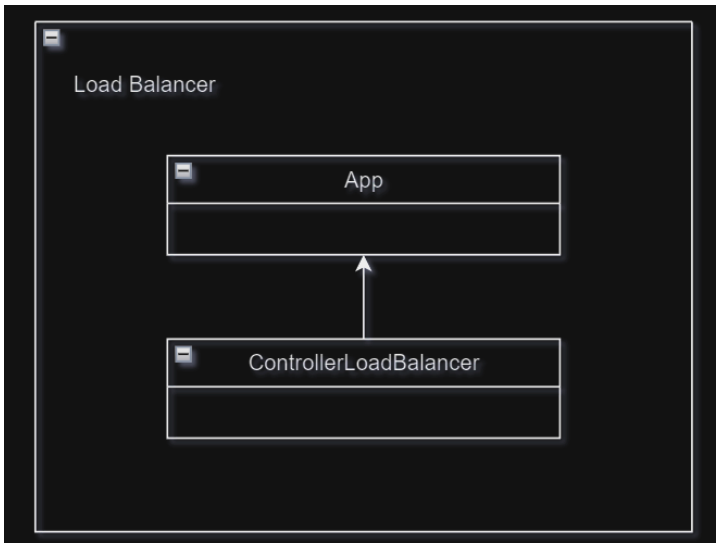
b. Diagrama de Clases:
    i. Models:



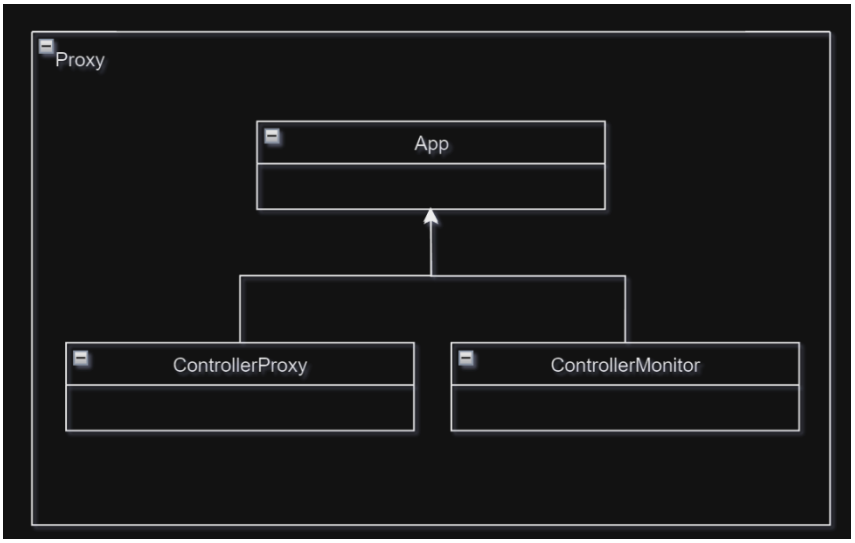
    ii. Edge: Sensor.



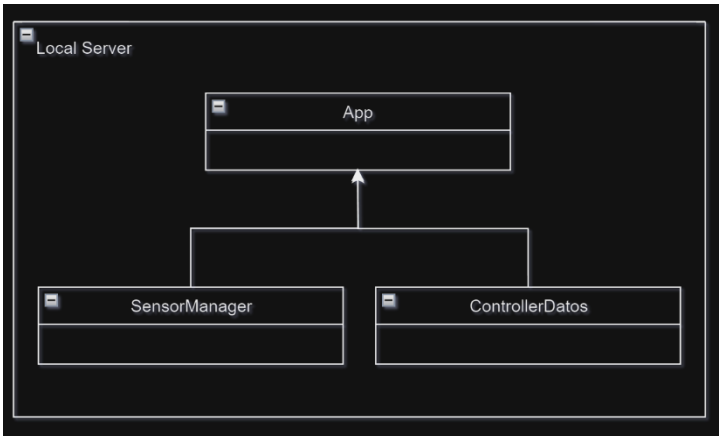
iii. Fog: Load Balancer.



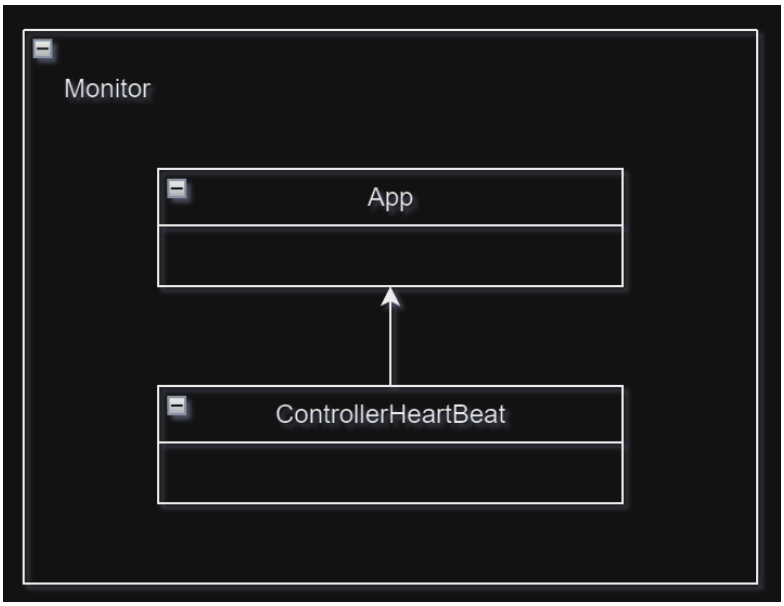
iv. Fog: Proxy.



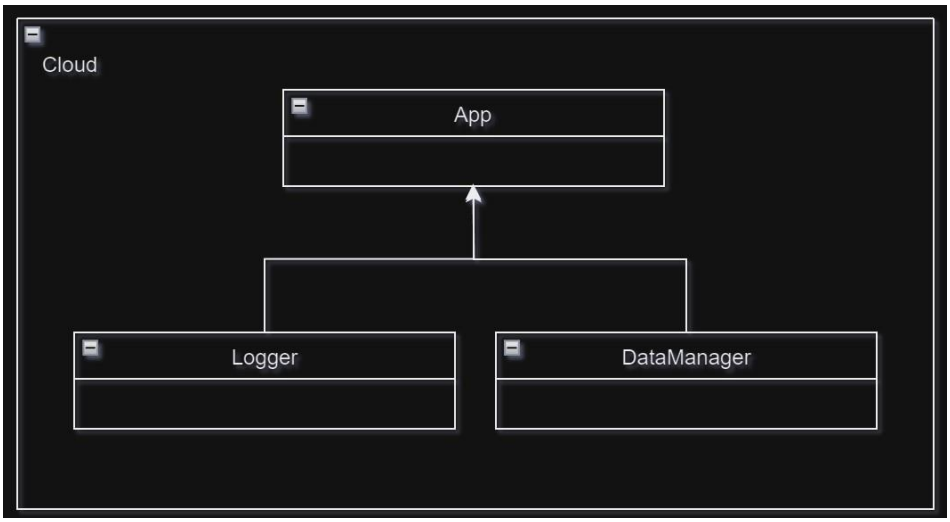
v. Fog: Local Server.



vi. Fog: Monitor

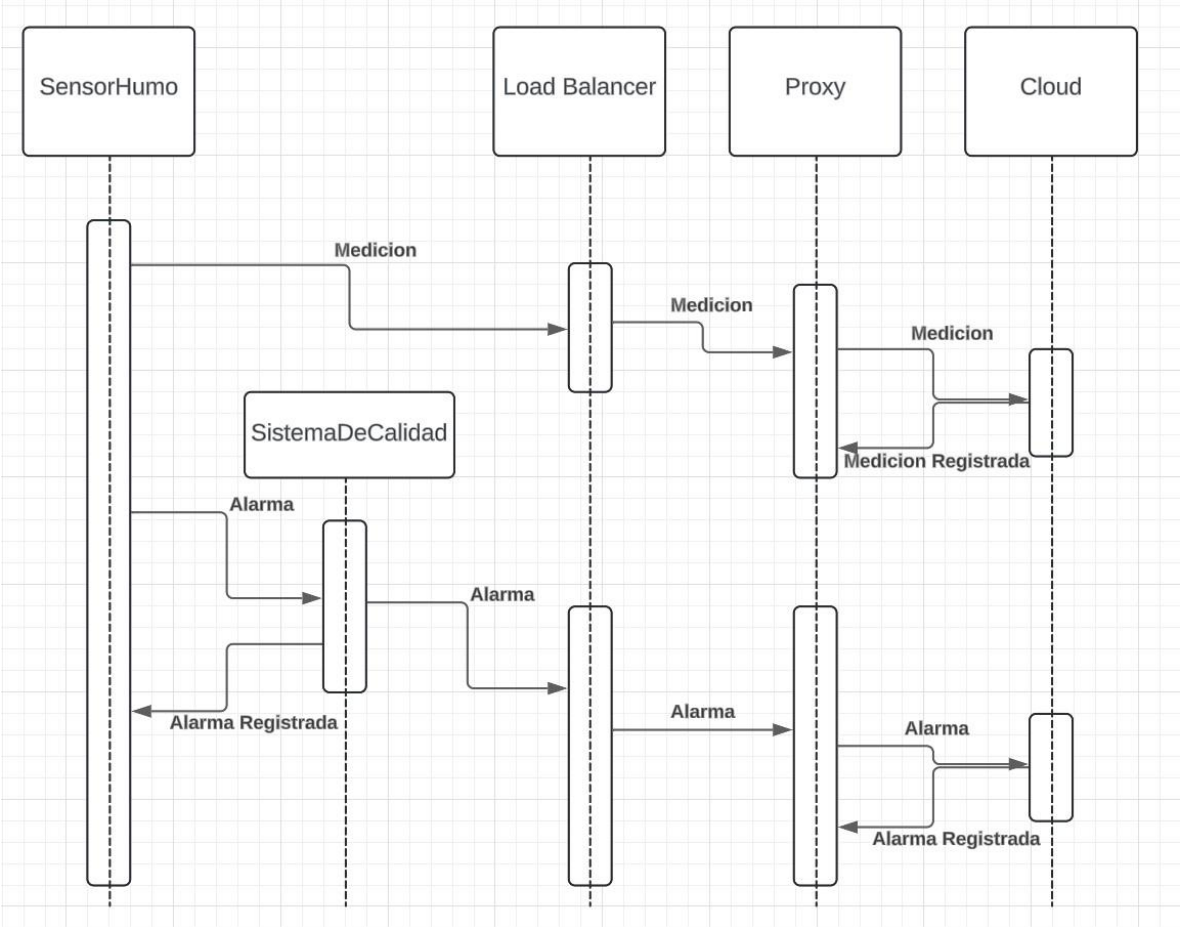


vii. Cloud



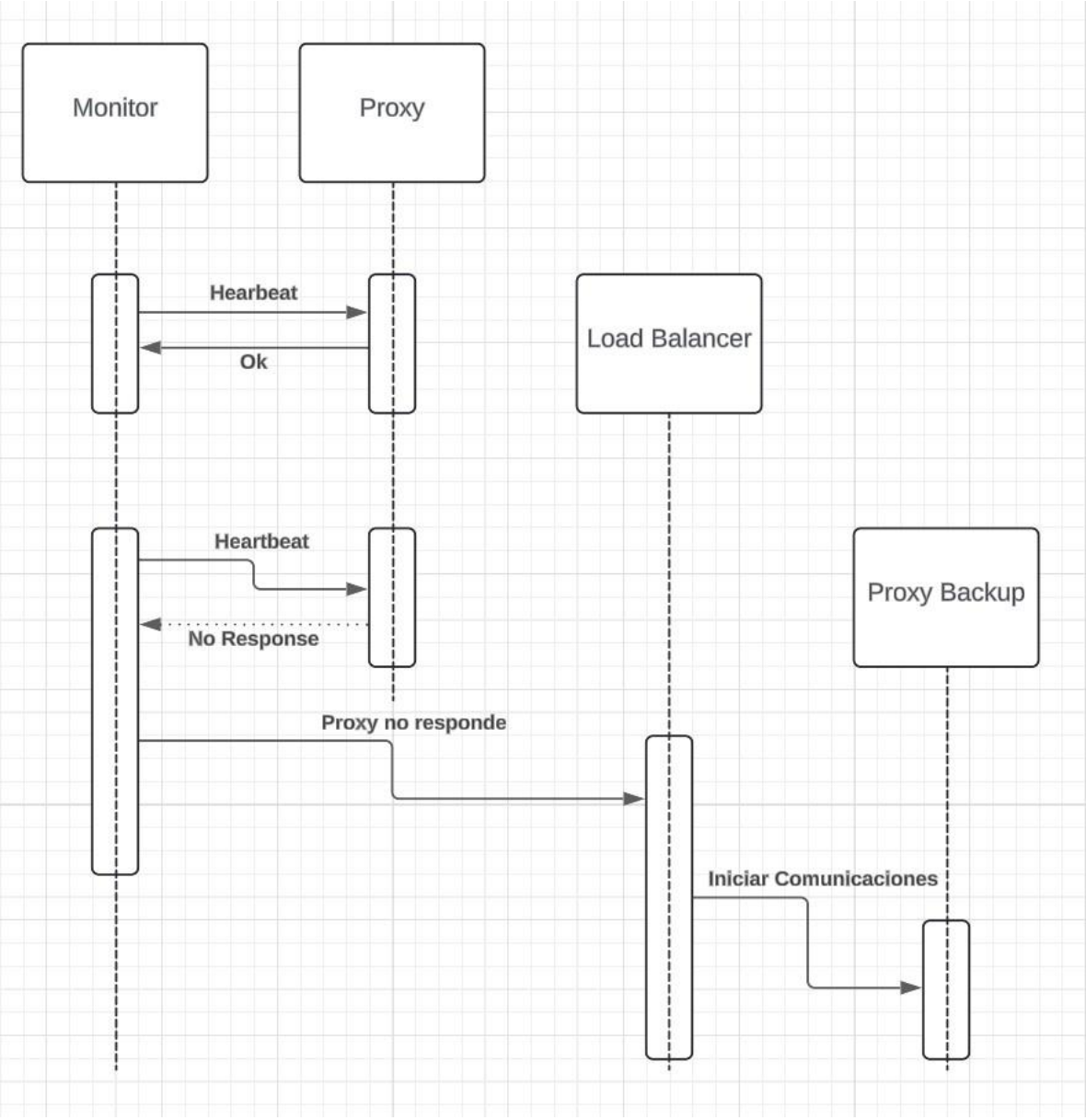
c. Diagrama de Secuencia:

i. Envío de Mediciones y Alarmas



4.

i. Fallo del Proxy



5. Métricas de Desempeño.

Para el apartado de métricas de desempeño haremos uso de la herramienta En el dinámico mundo del desarrollo de software, la optimización y el análisis profundo de las aplicaciones son aspectos cruciales para garantizar su correcto funcionamiento, escalabilidad y rendimiento. En este contexto, VisualVM emerge como una herramienta indispensable para desarrolladores y administradores de sistemas que trabajan con aplicaciones Java.

VisualVM se posiciona como una solución integral que va más allá del simple monitoreo, ofreciendo una amplia gama de funcionalidades que permiten observar el comportamiento de las aplicaciones Java en tiempo real y con gran detalle. A través de gráficos completos y análisis precisos, VisualVM brinda información valiosa sobre el uso de CPU, memoria (heap y no-heap), threads y clases cargadas, facilitando la comprensión del rendimiento de la aplicación y la identificación de posibles cuellos de botella.

Para optimizar aún más el rendimiento de las aplicaciones Java, VisualVM ofrece un análisis detallado de la actividad de la recolección de basura (GC). Esta información permite comprender cómo la JVM gestiona la memoria y realizar ajustes en la configuración para mejorar la eficiencia del proceso de GC.

La versatilidad de VisualVM se ve ampliada aún más gracias a su capacidad de ser extendida mediante plugins. Estos plugins añaden funcionalidades adicionales, como el monitoreo de aplicaciones JMX y el análisis de logs, convirtiéndola en una herramienta completa para el desarrollo y mantenimiento de aplicaciones Java.