Tarea 3 - INF326 - Arquitectura de Software

Pablo Arellano, ROL: **202073034-2**

Sebastian Donoso, ROL: **201921090-4**

Cristóbal Pérez: ROL: **202073105-5**

Enlace Repositorio: https://github.com/cristobalP02/tarea3inf326

Decisiones adecuadas y riesgosas

Decisiones adecuadas

Entrega de mensajes en dos etapas

Beneficia: Escalabilidad

Justificación: Al reducir el overhead del mensaje notificación de sismos, y permitir a los suscriptores consultar sobre datos de sismos específicos, permite que, al escalar, la latencia no se vea afectada por ciudades que no son afectadas por la alerta. Además, contribuye a optimizar el ancho de banda.

Decisiones riesgosas

Sistema de mensajería no redundante

Afecta: 99.9997% de disponibilidad.

Justificación: Un único componente publicador es un único punto de fallo (Single Point Of Failure). Si RabbitMQ falla, el sistema de notificación completo colapsa. Esto atenta directamente el concern de safety-critical y los requerimientos de disponibilidad solicitados.

Filtrado local de los sismos

Afecta: Escalabilidad

Justificación: Todos los suscriptores reciben notificaciones de los sismos ocurridos, aumentando el tráfico en la red de forma innecesaria mientras los suscriptores crecen.

Única localización física de despliegue de componentes críticos del sistema

Afecta: Safety-critical, Disponibilidad, Tolerancia a Fallos, Seguridad interna y externa.

Justificación: Desplegar los componentes críticos del sistema en un único lugar físico expuesto a desastres naturales, incendios o cortes de luz (provocados por el mismo sismo, por ejemplo) convierte a la localización en un *Single Point Of Failure*. De esta manera, un sismo se convierte en potencialmente en una de las vulnerabilidades del sistema. Además, si esta información se descubre por parte de atacantes, se puede realizar un cyber ataque en conjunto (como una denegación de servicios), o bien, un ataque físico.

Único servicio de endpoints disponible

Afecta: Safety-critical, Disponibilidad, Tolerancia a Fallos.

Justificación: Contar con un único servicio de endpoints provoca que, si el sismo es de importancia para múltiples suscriptores, las latencias aumenten. Además, un fallo en el sistema no detectado a tiempo no permitirá que los suscriptores accedan a información de los sismos cuando estos ocurran.

Identificación de puntos de sensibilidad

Message Broker (RabbitMQ)

Modificaciones en la forma que actúa RabbitMQ, como, agregar replicación de las colas, o incorporar un sistema de clustering, mejorará significativamente la disponibilidad y la tolerancia a fallos.

Redundancia de endpoints

La decisión de utilizar un único endpoint, o distribuido, tendrá un gran impacto en la disponibilidad del servicio.

Cuello de Botella

La decisión de utilizar solo un sistema de mensajería y un solo endpoint crea un cuello de botella, dado que el componente de mensajería en el Centro de Alertas es el que envía a cada nodo los mensajes y el endpoint de Datos de Sismos es el que recibe las peticiones de información de cada nodo que lo necesite. En el caso de aumentar nodos, aumenta la

cantidad de mensajes que debe enviar el sistema de mensajería y si hay más nodos, es más probable que el endpoint reciba más solicitudes.

Seguridad de mensajes

La seguridad ante potenciales ataques se debe concentrar en el Centro de Alarmas, la cola de mensajes y los nodos receptores de los mensajes. Se debe concentrar la seguridad del sistema de mensajería, para evitar manipulación de los mensajes, y la seguridad de la cola de mensajes, para asegurar que todos los nodos reciban correctamente los mensajes de alerta y asegurarse que ningún otro nodo que sea desconocido se pueda conectar a la cola.

Propuestas de decisiones mejoradas

Propuestas planteadas mejoradas

- Topología Broker distribuida y geo-redundante: El broker actual es un "cuello de botella", ya que, si falla, nada va a funcionar. Es por esto por lo que en vez de tener un solo "Message System", se duplique en varios servidores distribuidos a lo largo Chile (por ejemplo, uno en el norte, uno en el centro y otro en el sur). Así aumentaría la disponibilidad del sistema, ya que sería más resistente a terremotos y fallas eléctricas.
- Desplegar más componentes publisher: Si se incorporan múltiples broker de mensajería, se podría tener uno por zona del país, de forma que la latencia en la red sea mínima.
- Evitar alta latencia en el endpoint de Datos: Uno de los problemas al escalar el sistema, es que aumenta la cantidad de posibles nodos que cumplan el requisito para pedir más información al Centro de Alertas, específicamente el Endpoint de "Datos Sismos". Una forma de mejorar el rendimiento y reducir la carga del servicio, es manejar la concurrencia del endpoint utilizando procesamiento asincrónico para las peticiones HTTP.

Propuestas nuevas

- Seguridad en el envío y cola de mensajes: Una de las formas de aumentar la resistencia del sistema de mensajería frente a ataques, es administrar los nodos que pueden acceder a la suscripción de la cola, lo que reduce la posibilidad de ataques DoS (Denegación del Servicio) de suscriptores que aumentan la carga del sistema. También se evita que individuos no autorizados accedan a los mensajes y la información que contienen, lo que, además, puede ser logrado con una encriptación de los mensajes.
- Seguridad en el envío de mensajes: Por otro lado, también se debe evitar que un componente externo pueda enviar mensajes a la cola, esto se logra mediante la correcta autenticación en el Centro de Alertas.
- Caché local en los suscriptores: En cada región (subscriber), se guarda una copia temporal de los últimos sismos en una "memoria local". Si el "warning center" no responde, la región usa lo guardado en su memoria local para decidir si alertar, aunque sea con datos un poco viejos. Esto hace que el sistema sea más robusto, permitiendo alertas locales temporales y mejorando la tolerancia a fallos sin depender 100% del "warning center".