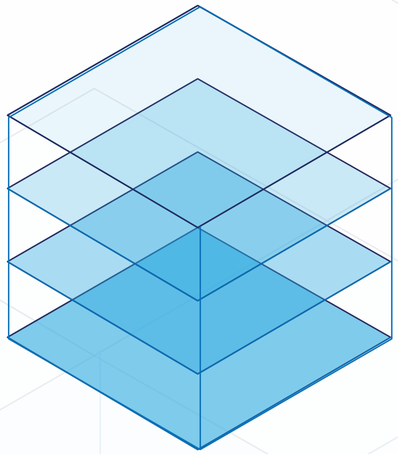


MISO

Maestría en Ingeniería de Software

Diseño de experimento Auditoria

Proyecto Final 1

Titulo del experimento	
Propósito del experimento	El objetivo del experimento es validar la capacidad del componente **Audit Service** para recibir y registrar, de manera asíncrona, todas las acciones críticas realizadas por el usuario desde otros componentes del sistema, como el Auth o el componente de gestión de incidentes. El punto de sensibilidad a experimentar es la capacidad del sistema para mantener la integridad y trazabilidad de los registros bajo condiciones de operación normal.
Resultados esperados	Se espera que el **Audit Service** registre correctamente el 100% de las acciones críticas enviadas por otros componentes, asegurando que cada acción esté asociada a un identificador único y contenga detalles relevantes para auditoría. Además, se debe verificar que el tiempo medio para revisar los registros de auditoría sea aceptable y que el registro se realice de manera eficiente sin afectar el rendimiento del sistema.
Recursos requeridos	• **Software:** NestJS para la implementación del backend, herramientas de pruebas como Postman para simular acciones de usuario, y una base de datos PostgreSQL para almacenamiento de registros. • **Librerías:** Librerías para soporte de comunicación asíncrona con PUB/SUB • **Hardware:** Instancias de CloudRun en la nube para desplegar el sistema y simular diferentes condiciones de carga.
Elementos de arquitectura invo lucrados	Componentes de **Auth**, **Incident Management**, y **Audit Service**. Ubicados en la vista de desarrollo en los modelos de componetes Seguridad, Busness Logic y Commons.
Esfuerzo estimado	Aproximamente 20 Horas/hombre distribuidas entre 4 ingenieros de desarrollo.

Hipótesis de diseño	
Punto de sensibilidad	Capacidad del **Audit Service** para registrar de manera asíncrona todas las acciones críticas del usuario provenientes de diferentes componentes del sistema (por ejemplo, Auth o Gestión de Incidentes). Esto incluye la habilidad de manejar la carga en picos de actividad y garantizar que el registro sea completo y sin pérdida de información.
Historia de arquitectura asociada	• **Como** Usuario, • **cuando** realizo una acción crítica como una transacción o cambio de configuración dado que estamos en operación normal, • **quiero** registrar todas las acciones críticas con identificadores únicos y detalles relevantes para asegurar la trazabilidad y auditoría. • **Esto debe suceder** 100% de acciones críticas registradas correctamente.
Nivel de incertidumbre	**Medio-Alto**: Existe incertidumbre sobre la capacidad del sistema para manejar eficientemente la carga de registros asíncronos sin perder acciones críticas, especialmente durante picos de actividad o bajo condiciones de alta concurrencia. También hay dudas sobre el rendimiento del sistema al manejar una gran cantidad de registros sin afectar negativamente otras operaciones del sistema.

Estilos de Arquitectura asociados al experimento	Análisis (Atributos de calidad que favorece y desfavorece)
El **Audit Service** se puede considerar un microservicio dedicado dentro de una arquitectura más grande basada en microservicios	**Favorece:** • **Escalabilidad:** La arquitectura basada en microservicios y eventos permite que el **Audit Service** escale independientemente para manejar un gran volumen de registros de auditoría, especialmente durante picos de actividad. • **Mantenibilidad:** Al ser un componente independiente en una arquitectura de microservicios, el **Audit Service** puede ser mantenido y actualizado sin afectar a otros componentes del sistema. **Desfavorece:** •**Consistencia eventual:** Dado que el registro de acciones críticas puede ser asíncrono, puede existir un periodo de tiempo en el que los datos de auditoría no reflejan inmediatamente las acciones más recientes. • **Latencia:** En una arquitectura basada en eventos, especialmente en sistemas distribuidos, puede haber una latencia adicional en el procesamiento de eventos y en la confirmación de que el registro de auditoría se ha realizado correctamente.

Tácticas de Arquitectura asociadas al experimento	Descripción
Seguridad: No Repudiation (No Repudio)	La táctica de **No Repudiation** (No Repudio) se enfoca en asegurar que todas las acciones realizadas por un usuario en el sistema sean rastreables de manera que el usuario no pueda negar su autoría. Para implementar esta táctica, es fundamental que el **Audit Service** registre cada acción crítica con un identificador único, una marca de tiempo, y detalles específicos sobre la operación realizada y el usuario responsable.
Seguridad: Audit (Auditoría)	La táctica de Audit implica la recopilación y el almacenamiento sistemático de información sobre las operaciones realizadas dentro del sistema para facilitar revisiones y auditorías futuras. Esta táctica es esencial para garantizar la transparencia y la responsabilidad en el manejo de datos y operaciones críticas. En el contexto del Audit Service, se establecerá un mecanismo de auditoría que registre automáticamente todas las acciones críticas (como transacciones financieras, cambios de configuración, accesos a datos sensibles, etc.) en una base de datos segura

Listado de componentes (Microservicios) involucrados en el experimento		
Microservicio	Propósito y comportamiento esperado	Tecnología Asociada
Audit Service	Registrar de manera asíncrona todas las acciones críticas realizadas por el usuario. Debe asegurar la trazabilidad y no repudio de cada acción registrada.	NestJS, PostgreSQL, pubsub
Auth Service	Enviar eventos al Audit Service cada vez que un usuario inicia sesión o realiza cambios en su configuración de acceso.	NestJS, PUBSUB
Incident Management Service	Reportar al Audit Service todas las modificaciones y accesos relacionados con la gestión de incidentes, asegurando que cada acción sea auditada.	NestJS, PUBSUB

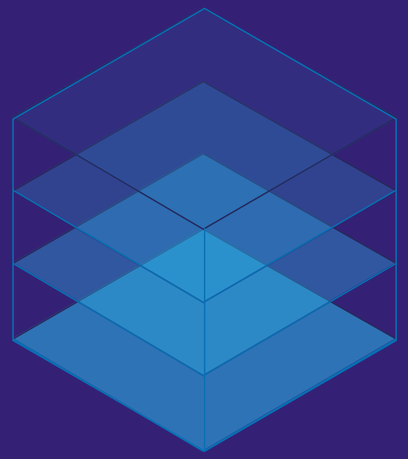
Listado de conectores involucrados en el experimento

Conector	Comportamiento deseado en el experimento	Tecnología Asociada
Audit pubsub susbcriber conector	Gestionar la comunicación asíncrona entre los microservicios y el **Audit Service** asegurando que los mensajes sean entregados de manera confiable y en el orden correcto.	Google cloud PubSub

Tecnología asociada con el experimento **(Desarrollo, infraestructura, almacenamiento)**	**Justificación**
Lenguajes de programación	**TypeScript**: Utilizado con **NestJS** para el desarrollo del backend, proporciona un tipado estático que mejora la robustez y mantenibilidad del código.
Plataforma de despliegue	**Google Cloud Run**: Permite desplegar contenedores en la nube con escalabilidad automática y gestión simplificada, ideal para microservicios desarrollados en NestJS.
Bases de datos	**PostgreSQL**: Elegido por su robustez, soporte de transacciones complejas, y capacidad para manejar un gran volumen de registros de auditoría de manera eficiente.
Herramientas de análisis	DataSpell para analizar la información registrada de eventos en la base de datos
Librerías	**Google Cloud Pub/Sub**: Utilizado para la mensajería asíncrona entre microservicios, permitiendo la publicación y suscripción de mensajes de manera eficiente y escalable.
Frameworks de desarrollo	**NestJS**: Un framework progresivo de Node.js que facilita la creación de aplicaciones escalables y mantenibles mediante el uso de patrones de diseño robustos y buenas prácticas.

Distribución de actividades por integrante

Integrante	Tareas a realizar	Esfuerzo Estimado
Daniel Jiménez	- Desarrollo del **Audit Service** en NestJS. - Desarrollo del **Auth Service** en NestJS. - Desarrollo del **Incident Manager Service** en NestJS. - Implementación de comunicación pub/sub entre los microservicios y el **Audit Service**.	20 h
Jesus Henrriquez	- Configuración de la infraestructura en **Google Cloud Run** para despliegue de microservicios. - Configuración de **Google Cloud Pub/Sub** para comunicación asíncrona. - Configuración de bases de datos en **PostgreSQL** para el almacenamiento de registros de auditoría.	5 h
Santiago Begambre	- Análisis de registros de auditoría utilizando **dataspell**. - Verificación de la integridad y consistencia de los registros de auditoría. - Identificación de patrones y posibles problemas en los registros de auditoría.	8 h
Carlos Castillo	- Preparacion del plan de pruebas y ejecucion de pruebas con postman runner para el consumo de operaciones del Auth Service y Audit Services - Generación de reportes detallados de los resultados del experimento. - Documentación de hallazgos y recomendaciones basadas en el análisis de datos.	8 h



MISO

Maestría en Ingeniería de Software