UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

Maestría en

Ingeniería de software

Asignatura:
Patrones de Integración Empresarial

Tarea Nro. 07:
Informe y Ejemplo de Uso: Filtro

Autor(es):
Cabascango Garcia Amanda Elizabeth
Calo Catota Carlos Edison
Fuentes Espinoza Pablo Gustavo
Guaman Guaman Saul German
Guerra Campuzano Cesar Hugo
Rengel Rivera Mateo Santiago
Vela Moya Christian Eduardo

Implementación de Filtros en la Arquitectura del Proyecto

Introducción

En el desarrollo de aplicaciones modernas, la implementación de filtros en diferentes niveles de la arquitectura es esencial para garantizar la seguridad, eficiencia y confiabilidad del sistema. Este informe analiza los distintos tipos de filtros implementados en el proyecto del Grupo 1, detallando su función y relevancia en la arquitectura general.

Marco Teórico

Filtros en Arquitectura de Software

Los filtros en una arquitectura de software permiten el control del flujo de datos entre distintos componentes del sistema. Se utilizan para asegurar que la información procesada cumpla con ciertos criterios antes de ser almacenada o enviada a otro servicio. Existen diferentes tipos de filtros dependiendo del contexto de aplicación:

- Filtros de Seguridad: Se emplean para verificar la autenticidad de los usuarios y restringir el acceso a servicios y datos sensibles.
- Filtros de Procesamiento de Datos: Garantizan que los datos sean válidos y cumplan con los requisitos de negocio antes de ser almacenados o consumidos por otros servicios.
- Filtros de Eventos y Mensajes: Permiten el procesamiento selectivo de eventos dentro de arquitecturas basadas en mensajería, asegurando que solo los eventos relevantes sean considerados.
- Filtros en el Flujo de Notificaciones: Controlan la distribución de notificaciones, garantizando que lleguen a los destinatarios adecuados.

- Filtros de Replicación de Datos: Aseguran que solo los datos necesarios sean replicados en bases de datos secundarias, optimizando el almacenamiento y la sincronización.

Importancia de los Filtros en Arquitecturas Distribuidas

En sistemas distribuidos, la implementación de filtros es crucial para mantener la seguridad, integridad y eficiencia de los datos. Sin ellos, la carga en los servidores y la red aumentaría significativamente, afectando la escalabilidad y el rendimiento del sistema. Entre los principales beneficios se encuentran:

- Reducción de carga en servidores: Al filtrar solicitudes innecesarias, se optimiza el procesamiento de datos.
- Mayor seguridad: Se evitan accesos no autorizados y se protegen datos sensibles.
- Eficiencia en la transferencia de datos: Se minimiza el tráfico de red al enviar solo la información requerida.

Filtros de Seguridad

Auth0

Auth0 es una plataforma de autenticación y autorización que actúa como un filtro de seguridad para los usuarios de la aplicación Postly App Flutter. Su función principal es garantizar que solo los usuarios autenticados puedan acceder a los servicios a través de GraphQL API. Auth0 ofrece una solución flexible y adaptable, permitiendo a los desarrolladores integrar servicios de autenticación y autorización de manera eficiente en sus aplicaciones. (auth0.com)

Implementación en el Proyecto:

- Registro y Configuración: Los usuarios se registran y autentican a través de Auth0, que gestiona las credenciales y tokens de acceso.
- Integración con GraphQL API: Auth0 se integra con GraphQL API para asegurar que solo las solicitudes de usuarios autenticados sean procesadas.

Kong Proxy Reverso

Kong se utiliza como un proxy reverso en la arquitectura para gestionar el enrutamiento de solicitudes hacia los servicios correspondientes. Aunque no se encarga directamente de la autenticación y autorización de tokens, desempeña un papel fundamental en la gestión y seguridad del tráfico de la API.

Funciones de Kong en el Proyecto:

- Balanceo de Carga: Distribuye el tráfico de red entre los diferentes servicios para optimizar el rendimiento.
- Enrutamiento de Solicitudes: Redirige las peticiones a los servicios apropiados según las reglas configuradas.
- Protección contra Ataques: Puede implementar medidas de seguridad como limitación de peticiones y detección de ataques DDoS.

Autenticación y Autorización de Tokens en GraphQL API

La verificación de la validez de los tokens JWT emitidos por Auth0 se realiza a nivel de GraphQL API, asegurando que solo usuarios autorizados accedan a los servicios. Esta implementación garantiza que cada petición sea validada correctamente antes de ser procesada.

Beneficios de la Implementación en GraphQL API:

- Seguridad Centralizada: Toda la autenticación y autorización se maneja en un solo punto.
- Mayor Flexibilidad: Permite implementar lógicas de autorización personalizadas según las necesidades del negocio.
- Eficiencia en el Procesamiento: Se evita el procesamiento innecesario de solicitudes no autorizadas desde el inicio.

Filtros de Procesamiento de Datos

GraphQL API

GraphQL es un lenguaje de consulta para APIs que permite a los clientes solicitar exactamente los datos que necesitan. En este proyecto, actúa como un filtro de datos, optimizando las respuestas y evitando la sobrecarga de información.

Beneficios:

- **Consultas Precisas:** Los clientes pueden especificar exactamente qué datos requieren, reduciendo el volumen de información transferida.
- **Eficiencia en el Uso de Recursos:** Al limitar los datos solicitados, se optimiza el rendimiento del servidor y se mejora la experiencia del usuario.

Servicios de Publicación y Comentarios

Los servicios de Post Service y Comment Service son responsables de gestionar las publicaciones y comentarios en la plataforma. Implementan filtros de negocio para validar y procesar los datos antes de su almacenamiento.

Funciones de los Filtros:

- Validación de Datos: Aseguran que las publicaciones y comentarios cumplan con los criterios establecidos (por ejemplo, longitud mínima, ausencia de contenido inapropiado).
- Enriquecimiento de Contenido: Pueden agregar metadatos o formatear el contenido según las necesidades de la aplicación.

Filtros de Eventos y Mensajes

Kafka Service

Apache Kafka es una plataforma de streaming de eventos distribuida que permite el procesamiento de flujos de datos en tiempo real. En este proyecto, Kafka actúa como un filtro de eventos, asegurando que solo los eventos relevantes sean procesados por los servicios correspondientes.

Uso en el Proyecto:

- Gestión de Eventos: Procesa eventos como "Nuevo Comentario" o "Actualización de Token FCM", distribuyéndolos a los servicios interesados.
- Filtrado de Eventos: Asegura que cada servicio consuma únicamente los eventos que le conciernen, evitando procesamiento innecesario.

Debezium Service

Debezium es una plataforma de captura de datos de cambio (Change Data Capture - CDC) que monitorea las bases de datos y registra las modificaciones en tiempo real. En este proyecto, Debezium detecta cambios en la base de datos de publicaciones y emite eventos cuando se producen modificaciones relevantes.

Funciones Clave:

- Monitoreo de Cambios: Observa las operaciones de inserción, actualización y eliminación en la base de datos.
- Emisión de Eventos: Genera eventos que pueden ser consumidos por otros servicios para mantener la coherencia de los datos en todo el sistema.

Filtros en el Flujo de Notificaciones

Noti Service

El servicio de notificaciones es responsable de gestionar y enviar notificaciones push a los usuarios. Implementa filtros para procesar eventos y garantizar que las notificaciones lleguen a los destinatarios correctos.

Procesos Implementados:

- Filtrado de Eventos de Comentarios: Convierte eventos de nuevos comentarios en notificaciones push, informando a los usuarios interesados.
- Gestión de Tokens FCM: Utiliza tokens de Firebase Cloud Messaging para dirigir las notificaciones a los dispositivos adecuados, asegurando una comunicación efectiva.

Filtros de Replicación de Datos

CDC Service y Replica DB

La replicación de datos es crucial para garantizar la disponibilidad y redundancia de la información. En este proyecto, se implementa un servicio de CDC que filtra y replica los datos de la base de datos principal a una base de datos réplica.

Características:

- Filtrado de Datos: Selecciona únicamente los datos relevantes para la replicación, evitando la duplicación innecesaria.
- Sincronización en Tiempo Real: Mantiene la base de datos réplica actualizada con los cambios en la base de datos principal.

Conclusiones

- La implementación de filtros en la arquitectura del proyecto ha demostrado ser una estrategia efectiva para optimizar el rendimiento, la seguridad y la escalabilidad del sistema. Los filtros de seguridad, como la autenticación con Auth0 y el control de acceso en GraphQL API, han fortalecido la protección de datos y garantizado que solo los usuarios autorizados puedan interactuar con los servicios.
- Los filtros de procesamiento de datos han mejorado la eficiencia de la API GraphQL, reduciendo la cantidad de datos transferidos y asegurando respuestas precisas a las solicitudes de los clientes. Asimismo, los filtros de eventos y mensajería han permitido una mejor gestión del flujo de datos en tiempo real, asegurando que los eventos sean procesados solo por los servicios pertinentes.
- En términos de notificaciones y replicación de datos, los filtros implementados han permitido que las notificaciones sean entregadas de manera eficiente y que los datos sean replicados sin redundancia innecesaria, asegurando la integridad y disponibilidad de la información en distintas bases de datos.
- En general, los filtros han permitido mejorar la eficiencia operativa del sistema, reduciendo la carga en los servidores y optimizando la distribución de datos en un entorno distribuido.

Recomendaciones

- Monitoreo y Optimización Continua: Se recomienda establecer un sistema de monitoreo que permita evaluar el rendimiento de los filtros implementados y realizar ajustes en función de la evolución del sistema y el crecimiento de la demanda.
- Ampliación de Políticas de Seguridad: Es aconsejable fortalecer la autenticación y autorización mediante la implementación de autenticación multifactor (MFA) y mecanismos avanzados de detección de anomalías para reforzar la seguridad del acceso a los servicios.
- Optimización de Consultas en GraphQL: Para mejorar aún más el rendimiento, se recomienda analizar y optimizar las consultas GraphQL, utilizando mecanismos de caché y limitación de consultas para evitar sobrecarga de datos.
- Estandarización de los Formatos de Datos: La implementación de un modelo de datos canónico permitiría mejorar la interoperabilidad entre servicios y reducir la necesidad de transformaciones adicionales en los filtros de datos.
- Automatización de la Gestión de Eventos: Se sugiere implementar herramientas de orquestación de eventos para garantizar que los filtros de mensajería operen de manera eficiente y permitan la escalabilidad del sistema en escenarios de alta concurrencia.
- Auditoría y Registro de Eventos: Se recomienda la implementación de registros de auditoría en los filtros de seguridad y procesamiento de datos para mejorar la trazabilidad y permitir un mejor análisis forense en caso de incidentes.

Bibliografía

- Auth0. (2024). *Getting Started with Auth0*. Recuperado de: https://auth0.com/docs/get-started/auth0-overview
- Red Hat. (2024). *Introduction to Debezium and CDC*. Recuperado de: https://debezium.io/documentation/
- Kong Inc. (2024). *API Gateway & Reverse Proxy*. Recuperado de: https://konghq.com/kong
- GraphQL Foundation. (2024). *Introduction to GraphQL*. Recuperado de: https://graphql.org/learn/