Estudio de caso

Vanguard

Del sistema monolítico a los microservicios: La mejora de la calidad del código mediante tecnología de grafos

INDUSTRIA
Servicios financieros

CASO DE USO
Modernización de aplicaciones

OBJETIVO
Reaccionar con rapidez para competir eficazmente, evitar disrupciones y cumplir con las normativas

RETO
Refactorizar a microservicios una descomunal base de código de producción

SOLUCIÓN
Uso de Neo4j para capturar y analizar todas las relaciones en el código heredado y los microservicios

RESULTADOS
- Comunicación y mejora de métricas de calidad de código objetivas
- Reducción de riesgos, realización de análisis de impacto e implementación de mejores prácticas

Vanguard Group tenía que refactorizar una maraña de código heredado esencial a microservicios, con el consiguiente aumento de partes móviles. Al capturar todas las dependencias en el grafo, el equipo racionaliza progresivamente su arquitectura e impulsa la refactorización, colaborando a nivel de empresa para mejorar las métricas de calidad del código derivadas del análisis de grafos.

La empresa

Vanguard es una de las compañías de inversión de capital más grandes del mundo, con más de 3,5 billones de dólares en activos bajo gestión. Es el mayor proveedor de fondos de inversión y el segundo proveedor de fondos cotizados (ETF). Con sus 17 600 empleados, Vanguard presta servicios a más de 20 millones de inversores en unos 170 países. Su fundador, John Bogle, es el creador de los fondos indexados.

El reto

Las empresas de servicios financieros requieren una avanzada infraestructura informática. Para refactorizar a infinidad de microservicios los sistemas monolíticos existentes basados en Java, Vanguard necesitaba la visibilidad de todos los componentes y sus dependencias. Algunos de los archivos Java (JAR) de Vanguard tienen de **3 a 4 millones de líneas de código**.

Además, Vanguard se enfrentaba a una deuda técnica. Tenían que eliminar código muerto y las evaluaciones de impacto eran difíciles.

„La gestión del código es crucial", dice John Lavin, arquitecto de software de Vanguard. „A menudo, la atención se centra en el suministro de funciones, el tiempo de comercialización y la clausura de un proyecto. Pero sin facilitar la gestión del código, no se consigue nada de eso."

En primer lugar, el equipo trató de visualizar las dependencias de los JAR en una herramienta de escritorio. Los intentos de cargar todos los JAR bloquearon la herramienta, que no estaba hecha para la envergadura de Vanguard.

Para refactorizar a microservicios archivos JAR con un millón de líneas de código hace falta gestionar muchas partes móviles. Con ese fin, el equipo empezó por gestionar los servicios en una hoja de cálculo, compilada en el transcurso de un año.

„Teníamos muchos servicios diferentes en muchos estados. Intentamos agrupar las cosas y documentar las dependencias lo mejor que pudimos en la hoja de cálculo. Pero estaba claro que no iba a funcionar a largo plazo", comenta Lavin.

Vanguard necesitaba modelar todos sus servicios y dependencias en el grafo para realizar análisis de impacto que mostraran cuántos servicios más se verían afectados si un determinado servicio no estuviera disponible. Las puertas de enlace de API que usan el tiempo de ejecución para capturar dependencias no funcionarían en el caso de Vanguard, porque había servicios clave solo utilizados en ciertas épocas del año y se omitirían.

Estudio de caso

„Los responsables van directamente a los paneles de información para ver si sus métricas tienden a subir o a bajar. La tecnología de grafos es excelente para incrementar métricas y mantener limpio el código, porque lo estamos observando.“

– John Lavin
Arquitecto de software,
Vanguard Group

La solución

El equipo necesitaba una solución flexible que pudiera adaptarse a niveles sin precedentes. “Nos dimos cuenta de que la gestión de nuestros módulos y servicios era realmente un problema de grafos”, admite Lavin. Vanguard adoptó Neo4j por su escala y su flexibilidad.

El equipo empezó con un modelo de datos de grafo simple, al que se iban agregando dependencias JAR automáticamente cuando ejecutaban una compilación. Luego agregaron al grafo todos los artefactos del código existente, junto con sus dependencias.

Mediante el análisis de grafos, el equipo empezó a evaluar su código con relación a las mejores prácticas y a derivar métricas. Luego agregaron información de su hoja de cálculo de arquitectura para enriquecer su esquema.

El equipo creó herramientas para visualizar las relaciones y aplicar mejores prácticas tales como restringir la cantidad de llamadas de servicio a servicio para reducir riesgos. Basándose en métricas derivadas del grafo, un cuadro de mandos de calidad de código permite la colaboración en toda la empresa, desde el nivel ejecutivo a los desarrolladores.

Los resultados

Vanguard consiguió una visibilidad completa de las dependencias entre archivos JAR existentes y microservicios, lo que les permitió realizar análisis de impacto y modernizar sus sistemas monolíticos de forma incremental.

Con todas las relaciones y dependencias capturadas en Neo4j, el equipo está abordando con eficacia la deuda técnica y demostrando mejoras concretas.

“Con métricas simples y unos cuantos cálculos, es posible obtener una gran cantidad de métricas de módulos utilizando consultas Cypher simples”, dice Lavin.

Vanguard visualiza las métricas derivadas de Neo4j en un cuadro de mandos de calidad de código que califica cada módulo. El cuadro de mandos se comparte con la empresa, junto con una visualización en forma de hoja de cálculo del grafo subyacente. La empresa ve tendencias que le sirven para medir el progreso de la iniciativa de modernización de aplicaciones.

“Los responsables van directamente a los paneles de información para ver si sus métricas tienden a subir o a bajar.

La tecnología de grafos es excelente para incrementar métricas y mantener limpio el código, porque lo estamos observando.”

Los arquitectos usan Neo4j no solo para gestionar el estado actual de todo el código de Vanguard sino también para avanzar hacia el estado deseado, maximizando los esfuerzos de los desarrolladores y promoviendo la reutilización.

“Cuando un equipo de proyecto entra en su fase de planificación, les proporcionamos información sobre qué servicios usar, dónde se están utilizando esos servicios, qué se ha creado parcialmente y dónde pueden agregar su lógica los desarrolladores para que tenga sentido en nuestra arquitectura general”, explica Lavin.

Neo4j es la plataforma líder en bases de datos de grafos que impulsa la innovación y la ventaja competitiva en empresas como Airbus, Comcast, eBay, NASA, UBS o Walmart. Miles de implementaciones de comunidad y más de 300 clientes utilizan Neo4j para aprovechar datos conectados que revelan interrelaciones entre personas, procesos, ubicaciones y sistemas. Mediante este enfoque basado en las relaciones, las aplicaciones basadas en Neo4j abordan los retos de los datos conectados, como la inteligencia artificial, la detección de fraudes, las recomendaciones en tiempo real y los datos maestros. Más información en Neo4j.com.

¿Tiene preguntas sobre Neo4j?
Contacte con nosotros
en todo el mundo:
info@neo4j.com
neo4j.com/contact-us