MICROPROFILE™

HISTORIA Y ORIGEN DE MICROPROFILE

MicroProfile nació en 2016 como una solución ligera y ágil frente a la lentitud de Java EE (hoy Jakarta EE) para adaptarse a los microservicios. Surgió de la comunidad Java para facilitar el desarrollo moderno de aplicaciones empresariales.

Inicialmente usó tecnologías conocidas como CDI, JAX-RS y JSON-P. Luego se integró a la Eclipse Foundation, ganando mayor respaldo y estructura. Con el tiempo, incorporó funcionalidades clave como configuración dinámica, métricas, OpenAPI y trazabilidad.

Hoy, MicroProfile es una alternativa robusta para crear microservicios en Java de forma eficiente y sin dependencias pesadas.

ARQUITECTURA DE MICROPROFILE

MicroProfile fue diseñado para facilitar el desarrollo de microservicios en Java, apoyándose en varios componentes de arquitectura esenciales:

- CDI: Gestión del ciclo de vida de objetos e inyección de dependencias. Promueve aplicaciones modulares y desacopladas.
- JAX-RS: Creación sencilla y estándar de APIs RESTful.
- JSON-P: Manejo manual de datos en formato JSON (lectura y escritura).
- JSON-B: Conversión automática entre objetos Java y JSON.

CARACTERISTICAS DE MICROPROFILE

MicroProfile es una iniciativa open source que adapta Java Empresarial a entornos modernos y nativos de la nube, facilitando el desarrollo de microservicios robustos.

Características destacadas:

Configuración externa: Adapta la app a distintos entornos sin recompilar.
Tolerancia a fallos: Maneja errores con reintentos, circuit breakers y fallbacks.
OpenAPI: Genera documentación clara e interactiva para APIs REST.
JWT Auth: Seguridad entre microservicios mediante tokens JWT.
Métricas: Monitorea el rendimiento y estado de la app en tiempo real.
Health Check: Verifica la salud de los servicios para detectar fallos tempranamente.

VENTAJAS DE MICRPROFILE

•**Estandarización:** Respaldado por grandes organizaciones (IBM, Red Hat, Apache), lo que garantiza consistencia y colaboración.

•**Flexibilidad:** No depende de un servidor específico; se puede ejecutar como JAR independiente o en servidores compatibles.

•**Integración moderna:** Compatible con Docker y Kubernetes para facilitar despliegues en contenedores.

•**Comunidad activa:** Mejora continua gracias al apoyo de una comunidad en crecimiento.

DESVENTAJAS DE MICRPROFILE

- **Madurez reciente:** A pesar del uso de tecnologías sólidas, la especificación como tal es relativamente nueva.

- **Documentación en español limitada:** La mayoría de los recursos están en inglés.

- **Curva de aprendizaje:** Requiere conocimientos en DI, AOP y servicios RESTful, lo que puede ser complejo para principiantes.

LENGUAJE ASOCIADO A MICROPROFILE

Lenguaje principal: *Java*
MicroProfile está diseñado específicamente para el desarrollo de microservicios en Java.

Relación con Java:

- Nació como una extensión ligera de **Jakarta EE** (antes Java EE).
- Utiliza **APIs estándar** como JAX-RS, CDI y JSON-P/JSON-B.
- Proporciona herramientas optimizadas para arquitecturas de **microservicios en Java**.
- Compatible con servidores Java como **Quarkus, Open Liberty, Payara Micro y Helidon**.

EJEMPLOS DE USO DE MICROPROFILE EN PROYECTOS REALES

•Helidon: Bibliotecas Java para crear microservicios. Implementa MicroProfile para facilitar su desarrollo.

•Apache TomEE: Servidor que combina Tomcat con tecnologías Java EE. Compatible con MicroProfile para microservicios ligeros.

•IBM Cloud & Open Liberty:IBM usa MicroProfile en Open Liberty para aplicaciones empresariales en la nube.
Ejemplo: Servicios bancarios con alta tolerancia a fallos.

•Mercado Libre: Usa Open Liberty + MicroProfile Metrics para autoescalado en Kubernetes durante eventos de alto tráfico como *Black Friday*.

CASOS DE USO COMUNES DE MICROPROFILE

- **Desarrollo de microservicios Java** en entornos empresariales.
- **Aplicaciones nativas para la nube**, integradas con Docker y Kubernetes.
- **Sistemas distribuidos** que requieren resiliencia, monitoreo y escalabilidad.
- **Servicios RESTful** que necesitan seguridad, documentación y configuración externa.
- **Aplicaciones que migran desde Java EE/Jakarta EE** hacia arquitecturas más ligeras.

RECOMENDACIONES PARA DESARROLLADORES QUE QUIERAN USAR MICROPROFILE

- Conocer Jakarta EE: Facilita el aprendizaje, ya que muchas APIs provienen de allí.
- Comenzar con una implementación ligera: Como Quarkus, Helidon o Open Liberty.
- Aprovechar la comunidad: Explorar foros, documentación oficial y ejemplos reales.
- Integrar herramientas modernas: Usar Docker, Kubernetes y CI/CD desde el inicio.
- Adoptar buenas prácticas: Como la inyección de dependencias, tolerancia a fallos y chequeo de salud.