Monitoreo y Trazabilidad

Plataforma de Servicios Estudiantiles

Proyecto: Progreso 2 - Integración de Sistemas
Estudiante: Martin Zumarraga
Profesor: Darío Villamarin G.
Fecha: 29 de Mayo, 2025

Objetivo del Documento

Este documento responde específicamente al **Requerimiento 5** del proyecto, explicando la implementación de monitoreo y trazabilidad en la arquitectura de integración de sistemas para la plataforma de servicios estudiantiles.

¿Qué herramientas utilizarías?

Stack de Herramientas Implementado

Zipkin - Trazas Distribuidas

- **URL:** http://localhost:9411
- **Función:** Seguimiento end-to-end de requests
- **Integración:** Spring Cloud Sleuth automático
- **Beneficios:**
  - Identificación rápida de cuellos de botella
  - Análisis de dependencias entre servicios
  - Debug de problemas de latencia

Prometheus - Métricas

- **URL:** http://localhost:9090
- **Función:** Colección y almacenamiento de métricas
- **Configuración:** Scraping del microservicio cada 5 segundos
- **Beneficios:**
  - Time-series database robusto
  - Query language potente (PromQL)
  - Integración nativa con Grafana

Grafana - Visualización

- **URL:** http://localhost:3000 (admin/admin)
- **Función:** Dashboards y alerting

- **Configuración:** Dashboards preconfigurados para métricas clave
- **Beneficios:**
    - Visualizaciones ricas e interactivas
    - Sistema de alertas flexible
    - Multi-datasource support

Spring Boot Actuator - Instrumentación

- **Endpoints:** `/actuator/health`, `/actuator/prometheus`, `/actuator/metrics`
- **Función:** Exposición de métricas y health checks
- **Configuración:** Habilitado para todos los endpoints de management
- **Beneficios:**
    - Instrumentación automática de Spring Boot
    - Métricas de JVM out-of-the-box
    - Health checks personalizables

Spring Cloud Sleuth - Instrumentación Automática

- **Función:** Propagación automática de contexto de trazas
- **Integración:** Transparente con Zipkin
- **Configuración:** Headers automáticos de correlación
- **Beneficios:**
    - Zero-code instrumentation
    - Integración con múltiples sistemas de tracing
    - Sampling configurable

Justificación de Selección de Herramientas

Criterios de Evaluación:

1. **Madurez:** Stack probado en producción a gran escala
2. **Integración:** Compatibilidad nativa con Spring Boot
3. **Comunidad:** Amplio soporte y documentación
4. **Escalabilidad:** Capacidad de crecer con el sistema
5. **Costo:** Herramientas open source sin licensing

Alternativas Consideradas:

- **ELK Stack:** Para logging centralizado (futuro)
- **Jaeger:** Alternativa a Zipkin (similar funcionalidad)
- **New Relic/DataDog:** APM comercial (mayor costo)
- **Kubernetes metrics:** Para entornos containerizados

¿Qué métricas y trazas capturarías?

Métricas Implementadas

Métricas de Negocio

Solicitudes Académicas:

- `solicitudes_total{tipo="CERTIFICADO_NOTAS"}` - Total por tipo de certificado
- `solicitudes_por_estado{estado="PROCESADO"}` - Distribución por estado
- `tiempo_procesamiento_promedio` - Latencia de procesamiento de solicitudes
- `certificaciones_exitosas_total` - Rate de éxito de certificaciones SOAP

Usuarios y Autenticación:

- `jwt_validaciones_total{resultado="exitoso"}` - Validaciones JWT por resultado
- `usuarios_activos_por_hora` - Patrones de uso temporal
- `solicitudes_por_estudiante` - Distribución de carga por usuario

⚙️ **Métricas Técnicas**

Performance de Endpoints:

```
# Latencia de endpoints
http_request_duration_seconds{method="POST",uri="/solicitudes"}
http_request_duration_seconds{method="GET",uri="/solicitudes/{id}"}

# Throughput
http_requests_total{method="POST",status="200"}
http_requests_total{method="GET",status="404"}

# Error Rate
rate(http_requests_total{status=~"5.."}[5m])
```

JVM y Sistema:

- `jvm_memory_used_bytes{area="heap"}` - Uso de memoria heap
- `jvm_gc_collection_seconds_sum` - Tiempo de garbage collection
- `jvm_threads_current` - Threads activos
- `system_cpu_usage` - Utilización de CPU

Integraciones Externas:

- `soap_calls_total{service="certificacion"}` - Llamadas SOAP totales
- `soap_call_duration_seconds` - Latencia de llamadas SOAP
- `soap_failures_total{reason="timeout"}` - Fallos por tipo

Conclusiones

Beneficios Obtenidos

1. **Visibilidad Completa**: Capacidad de rastrear cualquier request desde cliente hasta base de datos

2. **Detección Proactiva**: Alertas automáticas antes de que usuarios sean afectados
3. **Debug Eficiente**: Tiempo de resolución de problemas reducido de horas a minutos
4. **Toma de Decisiones**: Datos cuantitativos para optimización de arquitectura
5. **Mejora Continua**: Métricas de negocio para evolución del producto

---

Documento de Monitoreo y Trazabilidad v1.0
Proyecto: Integración de Sistemas - Universidad
Fecha: Mayo 2025