Proyecto Final IngeSoft V

Descripción

Para este proyecto final, deberá implementar una arquitectura completa de microservicios utilizando prácticas modernas de DevOps, seguridad y observabilidad. Trabajará con el código disponible en: https://github.com/SelimHorri/ecommerce-microservice-backend-app/ implementando todos los microservicios de la arquitectura e integrándolos en un entorno de Kubernetes.

Requisitos Principales

1. Metodología Ágil y Estrategia de Branching (10%)

- Implementar una metodología ágil (Scrum o Kanban) para el desarrollo del proyecto
- Definir y documentar una estrategia de branching (GitFlow, GitHub Flow o similar)
- Utilizar un sistema de gestión de proyectos ágiles (Jira, Trello, GitHub Projects, etc.)
- Documentar sprints, historias de usuario y criterios de aceptación
- Realizar al menos 2 iteraciones completas durante el desarrollo

2. Infraestructura como Código con Terraform (20%)

- Configurar toda la infraestructura necesaria usando Terraform
- Implementar estructura modular
- Implementar configuración para múltiples ambientes (dev, stage, prod)
- Documentar la arquitectura de infraestructura con diagramas
- Implementar backend remoto para el estado de Terraform

3. Patrones de Diseño (10%)

- Identificar y documentar los patrones de diseño utilizados en la arquitectura existente
- Implementar o mejorar al menos tres patrones adicionales:
  - Un patrón de resiliencia (Circuit Breaker, Bulkhead, etc.)
  - Un patrón de configuración (External Configuration, Feature Toggle, etc.)
- Documentar los patrones implementados, su propósito y beneficios

4. CI/CD Avanzado (15%)

- Implementar pipelines completos de CI/CD (Jenkins, GitHub Actions o Azure DevOps)
- Configurar ambientes separados (dev, stage, prod) con promoción controlada
- Implementar SonarQube para análisis estático de código
- Implementar Trivy para escaneo de vulnerabilidades en contenedores
- Implementar versionado semántico automático
- Configurar notificaciones automáticas para fallos en la pipeline
- Implementar aprobaciones para despliegues a producción

5. Pruebas Completas (15%)

- Implementar pruebas unitarias para los microservicios
- Implementar pruebas de integración entre servicios relacionados
- Implementar pruebas E2E para flujos completos de usuario
- Implementar pruebas de rendimiento y estrés con Locust
- Implementar pruebas de seguridad (OWASP ZAP o similar)
- Generar informes de cobertura y calidad de pruebas
- Configurar ejecución automatizada en pipelines

6. Change Management y Release Notes (5%)

- Definir un proceso formal de Change Management
- Implementar generación automática de Release Notes

- Documentar planes de rollback
- Implementar sistema de etiquetado de releases

7. Observabilidad y Monitoreo (10%)

- Implementar stack de monitoreo con Prometheus y Grafana
- Configurar ELK Stack (Elasticsearch, Logstash, Kibana) para gestión de logs
- Implementar dashboards relevantes para cada servicio
- Configurar alertas para situaciones críticas
- Implementar tracing distribuido (Jaeger, Zipkin, etc.)
- Configurar health checks y readiness/liveness probes
- Implementar métricas de negocio además de métricas técnicas

8. Seguridad (5%)

- Implementar escaneo continuo de vulnerabilidades
- Implementar gestión segura de secretos
- Configurar RBAC para acceso a recursos
- Implementar TLS para servicios expuestos públicamente

9. Documentación y Presentación (10%)

- Documentación completa del proyecto
- Repositorio Git organizado
- Costos de infraestructura
- Manual de operaciones básico
- Video demostrativo del funcionamiento
- Presentación del proyecto

Bonificaciones

1. Implementación Multi-Cloud (5%)

- Desplegar la aplicación en al menos dos proveedores cloud diferentes

- Implementar estrategia de respaldo entre clouds
- Configurar balanceo de carga entre proveedores
- Documentar comparativas de rendimiento entre clouds

2. GitOps con ArgoCD o Flux (5%)

- Implementar despliegue continuo con ArgoCD o Flux
- Configurar sincronización automática desde repositorio Git
- Implementar Progressive Delivery
- Configurar notificaciones de despliegue
- Implementar rollbacks automáticos basados en métricas

3. Service Mesh (5%)

- Implementar Istio, Linkerd o similar
- Configurar mTLS entre todos los servicios
- Implementar traffic shifting para despliegues canary
- Visualizar el mesh con herramientas adecuadas
- Implementar circuit breakers y retry policies

4. Chaos Engineering (5%)

- Implementar Chaos Mesh, Litmus Chaos o similar
- Diseñar y documentar experimentos de caos
- Ejecutar pruebas de resiliencia en diferentes componentes
- Documentar resultados y mejoras implementadas
- Integrar aprendizajes en la arquitectura

5. FinOps (5%)

- Implementar monitoreo de costos para todos los recursos
- Configurar políticas de ahorro (spot instances, scale to zero)
- Implementar dashboards de costos y utilización
- Realizar análisis de optimización de costos

- Documentar estrategias implementadas y ahorros conseguidos

6. Autoscaling Avanzado con KEDA (5%)

- Implementar KEDA para autoscaling basado en eventos
- Configurar al menos dos triggers diferentes para autoscaling
- Demostrar y documentar el comportamiento del autoscaling

Entregables

1. Código fuente completo en repositorio Git
2. Documentación completa del proyecto incluyendo:
    - Arquitectura detallada con diagramas
    - Descripción de la metodología ágil implementada
    - Documentación de patrones de diseño
    - Guías de operación y mantenimiento
    - Análisis de resultados de pruebas
    - Documentación de la infraestructura como código
    - Release Notes de cada versión
3. Presentación y demostración (20-30 minutos) que incluya:
    - Arquitectura e infraestructura
    - Demostración de CI/CD
    - Demostración de la aplicación funcionando
    - Dashboards de monitoreo
    - Resultados de pruebas de rendimiento
    - Lecciones aprendidas y recomendaciones