**Prueba Técnica - Desarrollador Java Jr**

**API REST con Integración de Servicios Externos**

**Objetivo**

Desarrollar una API REST que consuma y procese datos de servicios externos.

**Instrucciones Generales**

* **Framework**: Quarkus es el framework preferido, sin embargo es aceptable utilizar el framework con el que se sienta más cómodo a fin de cumplir la entrega en el plazo establecido.
* **Entregable**: Repositorio Git con código fuente y README con instrucciones.

**API Externa a Utilizar**

Deberás consumir la API pública **JSONPlaceholder**: <https://jsonplaceholder.typicode.com/>

**Endpoints Requeridos**

**1. GET /posts (Endpoint Principal)**

Este endpoint debe realizar múltiples llamadas a la API externa, mergear la información y retornar datos procesados.

**Funcionalidad esperada:**

* Obtener la lista de posts desde https://jsonplaceholder.typicode.com/posts
* Para cada post, obtener los comentarios desde https://jsonplaceholder.typicode.com/posts/{postId}/comments
* Para cada post, obtener información del usuario autor desde https://jsonplaceholder.typicode.com/users/{userId}
* Mergear toda la información y retornar algunos valores de cada endpoint (o todos si se prefiere)

**2. DELETE /posts/{id} (Endpoint Secundario)**

* Realizar llamada DELETE a https://jsonplaceholder.typicode.com/posts/{id}
* Nota: La API no persiste cambios realmente, solo simula la operación
* Retornar respuesta HTTP apropiada

**Requisitos Técnicos**

**Obligatorios:**

* API REST funcional con los endpoints especificados
* Manejo de excepciones y errores HTTP
* Logging básico
* README con instrucciones de ejecución

**Valorables:**

* Tests unitarios para al menos el endpoint principal
* Configuración externalizada (application.properties/yaml)
* Implementación de cache para optimizar llamadas repetidas
* Documentación con OpenAPI/Swagger
* Validaciones de entrada y sanitización

**Criterios de Evaluación**

1. **Funcionalidad**: Los endpoints funcionan correctamente
2. **Integración**: Consumo adecuado de múltiples servicios externos
3. **Código**: Calidad, organización y buenas prácticas
4. **Manejo de errores**: Gestión apropiada de fallos y timeouts
5. **Testing**: Diseño
6. **Performance**: Eficiencia en las llamadas concurrentes

**Entrega**

* Subir código a repositorio Git (GitHub, GitLab, etc.)
* Incluir README detallado con:
  + Instrucciones de compilación y ejecución
  + Descripción de la arquitectura
  + Ejemplos de uso de los endpoints
  + Decisiones técnicas tomadas