PuntoNet API Microservicio start.gg – Argumento de Separación de Responsabilidades

1. Contexto

Actualmente, se tiene una aplicación frontend en Angular que se alimenta de una API REST en Spring Boot para registrar porcentajes de KO en un juego de peleas (tipo Super Smash o Fighting Platform Game). Por otro lado, start.gg ofrece una API REST que permite acceder a información detallada de asistencia de jugadores a torneos. Esta información es volumétrica y requiere procesamiento.

2. Problema

Integrar directamente la lógica de procesamiento de start.gg en la API de Spring Boot podría:

- Aumentar la complejidad de la API.
- Generar retrasos por la gran cantidad de datos.
- Limitar la reutilización de la lógica para otros posibles frontends.

3. Solución propuesta

Crear un microservicio independiente en .NET encargado de:

- Conectarse a la API de start.gg.
- Procesar y filtrar los datos (ej. jugadores en la región de Valparaíso, ordenados por asistencia mensual).
- Exponer sus propios endpoints REST para consumo del frontend o la API principal.

4. Ventajas

1. Separación de responsabilidades: Cada servicio tiene un objetivo claro y limitado.
2. Escalabilidad: El microservicio puede escalar de forma independiente para manejar grandes volúmenes de datos.
3. Reutilización: Otros frontends o servicios pueden consumir la API .NET sin afectar Spring Boot.
4. Flexibilidad tecnológica: Permite usar .NET por sus ventajas o librerías específicas.

5. Opciones de integración

- Angular consume directamente el microservicio .NET.
- Spring Boot actúa como intermediario y consolida datos de KO y start.gg antes de enviarlos al frontend.

6. Optimización

- Implementar cacheo (Redis, MemoryCache) para reducir latencia.
- Actualizar datos de start.gg de manera periódica en lugar de consultas en tiempo real constantes.

7. Conclusión

Separar el procesamiento de datos de start.gg en un microservicio .NET es una práctica válida, eficiente y alineada con buenas prácticas de arquitectura de software. Permite escalabilidad, mantenimiento y reutilización, evitando sobrecargar la API principal de Spring Boot.