

# Repositorio GitHub

https://github.com/vquescam/ProgramacionConcurrente/tree/main/Feedback%202

# Diseño de la Arquitectura Integrada del SMGA

El objetivo del sistema SMGA es integrar de manera robusta y escalable cuatro subsistemas críticos, cada uno especializado en un área diferente, garantizando interoperabilidad, resiliencia y capacidad de respuesta ante eventos en tiempo real.

## Subsistemas Integrados

Gestión de Pedidos en Tiempo Real

Tecnologías: Spring, HikariCP, @Transactional

Funcionalidad: Maneja transacciones concurrentes y gestiona el acceso a la base de datos de manera eficiente utilizando técnicas de connection pooling y configuraciones transaccionales robustas.

Monitoreo de Dinosaurios en Jurassic Park

Tecnologías: Programación reactiva con Spring WebFlux

Funcionalidad: Recibe y procesa grandes volúmenes de datos de sensores en tiempo real, aprovechando la arquitectura reactiva para gestionar la alta concurrencia y latencia mínima.

Misión en Marte

Tecnologías: Spring Batch

Funcionalidad: Procesa grandes volúmenes de datos de sensores en lotes, realizando análisis y transformaciones de los datos recolectados en intervalos definidos.

Monitoreo y Gestión de Hechizos en el Mundo Mágico

Tecnologías: Microservicios, Spring Cloud

Funcionalidad: Gestiona eventos mágicos con alta resiliencia, utilizando patrones de circuit breaker, discovery server y técnicas de balanceo de carga para asegurar la continuidad y eficiencia del servicio.

### Estrategia de Integración

Para integrar estos subsistemas en el SMGA se recomienda la siguiente estrategia:

API Gateway/Servicio de Integración:

Todos los subsistemas se exponen a través de un API Gateway que centraliza las solicitudes, permitirá la autenticación y enrutamiento a cada subsistema correspondiente.

Interoperabilidad a través de Mensajería Asíncrona:

Se implementa un sistema de mensajería (por ejemplo, RabbitMQ o Kafka) que permita la comunicación asíncrona entre subsistemas, facilitando la integración y mejorando la resiliencia ante fallos.

Gestión Centralizada de Configuraciones y Monitoreo:

Se utiliza una solución de configuración centralizada (por ejemplo, Spring Cloud Config) junto con herramientas de monitoreo y logging centralizadas (como ELK, Prometheus, y Grafana) para supervisar el estado de cada subsistema.

Resiliencia y Escalabilidad:

Circuit Breaker: Cada servicio empleará patrones de circuit breaker para aislar fallos y evitar que una falla en un subsistema afecte a toda la aplicación.

Balanceo de Carga: Infraestructura escalable que permita el balanceo de carga horizontal, adaptándose a picos de tráfico y asegurando la disponibilidad de los servicios.

Desacoplamiento: El uso de colas de mensajes entre servicios garantiza que cada subsistema opere de forma independiente, lo que permite mejoras y mantenimiento sin afectar la operatividad global.

### Diagrama de Arquitectura

A continuación se presenta un diagrama de la arquitectura integrada utilizando Mermaid para visualizar la relación entre los distintos subsistemas:

flowchart LR

subgraph SMGA [Sistema de Monitoreo y Gestión Avanzada (SMGA)]

A[API Gateway / Servicio de Integración]

B[Gestión de Pedidos en Tiempo Real]

C[Monitoreo de Dinosaurios en Jurassic Park]

D[Misión en Marte]

E[Monitoreo y Gestión de Hechizos en el Mundo Mágico]

end

subgraph Mensajería [Sistema de Mensajería Asíncrona]

MQ[Mensaje Broker (RabbitMQ / Kafka)]

end

%% Relación entre API Gateway y subsistemas

A --> B

A --> C

A --> D

A --> E

%% Comunicación asíncrona entre subsistemas (ej. para eventos críticos)

B -- Eventos de Pedidos --> MQ

C -- Datos de Sensores --> MQ

D -- Resultados de Procesamiento --> MQ

E -- Eventos Mágicos --> MQ

%% Configuración y Monitoreo Centralizado

CONFIG[Config Server (Spring Cloud Config)]

MONITOR[Monitoreo y Logging Central (ELK, Prometheus, Grafana)]

A --- CONFIG

A --- MONITOR

B --- CONFIG

C --- CONFIG

D --- CONFIG

E --- CONFIG

B --- MONITOR

C --- MONITOR

D --- MONITOR

E --- MONITOR

%% Resiliencia y Balanceo de Carga

LB[Balanceador de Carga]

LB --> A

# Desarrollo y Optimización del SMGA

Resuelto en el código del proyecto ubicado en el repositorio de GitHub https://github.com/vquescam/ProgramacionConcurrente/tree/main/Feedback%202

# Implementación de la integración

Resuelto en el código del proyecto ubicado en el repositorio de GitHub https://github.com/vquescam/ProgramacionConcurrente/tree/main/Feedback%202

# Pruebas y Validación del SMGA

Resuelto en el código del proyecto ubicado en el repositorio de GitHub https://github.com/vquescam/ProgramacionConcurrente/tree/main/Feedback%202

# Análisis Post-Implementación