Informe Técnico: Microservicio de Notificaciones "GastanGO-Notify"

Fecha: 26 de octubre de 2025

Equipo: Grupo 14 - Joel Tapia

1. Resumen del microservicio creado

Nombre del microservicio

GastanGO-Notify

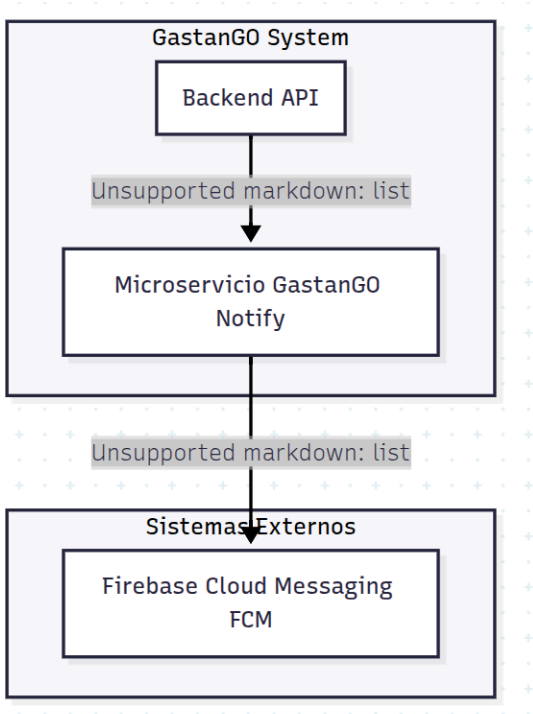
Propósito y Relación

GastanGO-Notify centraliza y desacopla la lógica de comunicación con el usuario, antes integrada en la API principal. Ahora, el Backend API solo solicita el envío de notificaciones (push, email o SMS), mientras GastanGO-Notify gestiona la integración con servicios externos como FCM.

Cuando el Backend API detecta un evento (ej. “transacción registrada” o “presupuesto excedido”), llama a GastanGO-Notify, que procesa y envía la notificación correspondiente.

Diagrama de Estructura (Mermaid)

Este diagrama ilustra la nueva relación. El Backend API ya no habla con FCM; ahora delega esa responsabilidad al nuevo microservicio.



2. Integración mediante APIs

Descripción del método de comunicación

La integración entre ambos servicios se realiza mediante una API REST síncrona (service-to-service).
Cuando el Backend API registra una transacción, envía una solicitud HTTP POST al endpoint de GastanGO-Notify para generar la notificación.
Aunque podría usarse un API Gateway como intermediario, en este caso una llamada directa es suficiente. La comunicación se asegura con una API Key o token M2M, restringiendo el acceso solo al Backend API.

Ejemplos de llamadas y respuestas en insomnia

Este es un ejemplo de la definición de la API para el nuevo microservicio.

Endpoint: **POST /api/v1/notify/push - Microservicio: GastanGO-Notify**

Propósito: Envía una notificación push simple a un usuario específico.

Ejemplo de Request (Llamada desde el Backend API):

```
{
  "userId": "usr_a1b2c3d4",
  "title": "¡Nuevo gasto registrado!",
  "body": "Se registró un gasto de $15.00 en 'Almuerzo'.",
  "data": {
    "screen": "TransactionDetail",
    "transactionId": "txn_xyz987"
  }
```

Ejemplo de Response (Respuesta de GastanGO-Notify):

Una buena práctica para servicios de notificación es responder rápido, indicando que el trabajo fue aceptado para procesamiento (HTTP 202), aunque el envío final ocurra en segundo plano.

```
/*
  HTTP/1.1 202 Accepted
*/
{
```

"status": "queued",

"message": "Notificación encolada para envío.",

"serviceMessageId": "fcm_msg_1a2b3c"

}

3. Herramientas colaborativas utilizadas

Para gestionar el desarrollo del microservicio GastanGO-Notify y su integración con el sistema principal, se empleó un flujo de trabajo basado en GitHub y GitKraken.

GitHub (Control de versiones y repositorios)

GitHub fue la herramienta principal para el control de versiones y gestión de repositorios.

Se creó un repositorio independiente llamado gastango-notify, separado de gastango-api, aplicando un flujo GitFlow simplificado con las ramas main y feature/notify-microservice.

Los avances se registraron con commits descriptivos, manteniendo trazabilidad y un historial limpio. Además, se integró gastango-api para consumir el endpoint POST /notify/push del nuevo microservicio.

GitKraken (Interfaz visual para Git)

GitKraken se utilizó como cliente visual para gestionar ramas, commits y fusiones de forma organizada.

Facilitó la visualización del historial, la resolución de conflictos y la integración directa con GitHub, permitiendo un control eficiente de versiones y una estructura de desarrollo clara incluso trabajando individualmente.

4. Buenas prácticas y aprendizajes

Principio de Responsabilidad Única (SRP): El Backend API se centra solo en la lógica de negocio (transacciones, presupuestos), mientras que GastanGO-Notify gestiona exclusivamente las notificaciones. Esto permite ampliar canales (Email con SendGrid, SMS con Twilio) sin modificar el Backend API.

Configuración externalizada (Variables de entorno): Las claves y URLs sensibles (como FCM_SERVER_KEY y la URL del microservicio) se gestionan mediante variables de entorno (.env), evitando valores hardcodeados y facilitando la configuración distinta para entornos de desarrollo y producción.

Reflexión personal

La modularización con microservicios mejora significativamente la escalabilidad y el mantenimiento del sistema.

Antes, el sistema monolítico se veía afectado por cualquier carga alta o proceso intensivo, como el envío masivo de notificaciones. Ahora, cada servicio escala de forma independiente; si las notificaciones generan alta demanda, se pueden desplegar más instancias de GastanGO-Notify sin afectar el Backend API.

En mantenimiento, la independencia entre servicios permite actualizaciones, correcciones y despliegues del módulo de notificaciones sin interrumpir el resto del sistema, reduciendo riesgos y tiempos de prueba.

Preguntas de reflexión:

¿Qué ventajas observas en la integración mediante APIs REST respecto a un monolito tradicional?

Las APIs REST permiten modularidad, escalabilidad y despliegue independiente, evitando que cambios en un servicio afecten todo el sistema, a diferencia del monolito.

¿Cómo aportan las herramientas colaborativas a la gestión técnica de los microservicios?

Herramientas como GitHub y GitKraken facilitan la colaboración, control de versiones y trazabilidad, asegurando una gestión ordenada y coherente entre microservicios.

¿Qué riesgos existen al distribuir un sistema en varios microservicios y cómo pueden mitigarse?

Los riesgos son la complejidad en la comunicación y la latencia; se mitigan con API Gateways, autenticación segura, monitoreo, logs centralizados y pruebas automatizadas.