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

El propósito de GastanGO-Notify es desacoplar y centralizar toda la lógica de comunicación con el usuario. Anteriormente, esta lógica residía dentro de la API principal (Backend API).

Al extraerlo, el Backend API (ahora un "core service") ya no necesita saber cómo se envía una notificación (si es por Firebase (FCM), email o SMS). Simplemente le pide a GastanGO-Notify que envíe un mensaje, y este nuevo microservicio se encarga de la integración con proveedores externos como FCM.

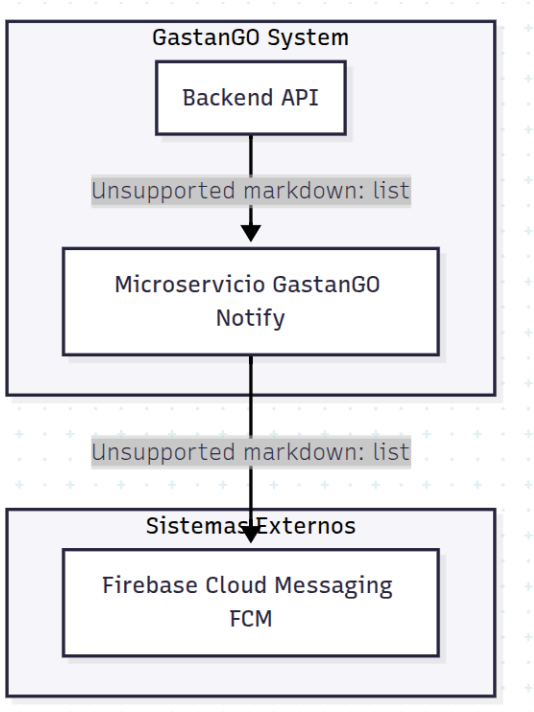
Relación con el sistema principal:

El Backend API (consumidor) llama a GastanGO-Notify (proveedor) cuando ocurre un evento (ej. "transacción registrada" o "presupuesto excedido").

GastanGO-Notify recibe la solicitud y la procesa (ej. enviando la push notification vía FCM).

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 se realiza mediante una API REST síncrona entre los dos servicios (comunicación service-to-service).

Cuando el Backend API necesita enviar una notificación (por ejemplo, después de que el TransactionController guarda una transacción), realiza una llamada HTTP POST al endpoint expuesto por GastanGO-Notify.

Para un sistema más robusto, se podría implementar un API Gateway que actúe como intermediario, pero para este caso de uso simple, una llamada directa es suficiente. La comunicación debe estar asegurada (ej. mediante una API Key interna o un token de servicio M2M) para que solo el Backend API pueda consumir este endpoint.

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 la gestión de los repositorios.

Repositorios: Se creó un nuevo repositorio independiente llamado gastango-notify para el microservicio, manteniéndolo separado del gastango-api.

Flujo de trabajo: Se aplicó un flujo GitFlow simplificado, trabajando principalmente con las ramas main y feature/notify-microservice.

Gestión de cambios: Cada avance significativo se registró mediante commits descriptivos, permitiendo llevar un control claro de la evolución del código.

Integración: Se realizaron ajustes en el gastango-api para que consumiera el nuevo microservicio a través del endpoint POST /notify/push.

Evidencia: GitHub permitió mantener la trazabilidad de los cambios y documentar las decisiones de integración mediante mensajes de commit detallados y un historial limpio.

GitKraken (Interfaz visual para Git)

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

Gestión visual: Facilitó la creación de ramas, la visualización del historial y la resolución de conflictos.

Automatización: Simplificó el flujo de trabajo al integrar de forma directa con GitHub.

Evidencia: Gracias a GitKraken, se logró mantener un control eficiente de las versiones y una estructura de desarrollo clara, incluso trabajando de forma individual.

4. Buenas prácticas y aprendizajes

Buenas prácticas aplicadas

Principio de Responsabilidad Única (SRP): Esta es la práctica fundamental aplicada. El Backend API ahora solo se enfoca en la lógica de negocio (CRUD de transacciones, presupuestos). El microservicio GastanGO-Notify tiene una sola responsabilidad: gestionar las comunicaciones con el usuario. Si mañana queremos añadir notificaciones por Email (ej. SendGrid) o SMS (ej. Twilio), solo modificamos GastanGO-Notify sin tocar el Backend API.

Configuración externalizada (Variables de Entorno): Las claves sensibles (como la FCM_SERVER_KEY) y las URLs de los servicios (la URL del nuevo microservicio) no están "hardcodeadas". Se gestionan a través de variables de entorno (.env), lo que permite que el Backend API llame a una URL de GastanGO-Notify en localhost durante el desarrollo y a una URL diferente en producción.

Reflexión personal

La modularización mediante microservicios impacta directamente en la escalabilidad y el mantenimiento.

En términos de escalabilidad, nuestro sistema anterior era un monolito. Si 1000 usuarios registraban gastos al mismo tiempo, y además teníamos que enviar 50,000 recordatorios de presupuestos, todo el Backend API se vería afectado. Ahora, podemos escalar los servicios de forma independiente. Si el envío de notificaciones se vuelve un cuello de botella, podemos desplegar 10 instancias de GastanGO-Notify sin necesidad de escalar el Backend API.

En cuanto al mantenimiento, la ventaja es la autonomía del equipo y la reducción del riesgo. Si el equipo de notificaciones (incluso si es una sola persona) necesita actualizar la API de FCM o corregir un bug en la lógica de envío, puede desplegar una nueva versión de GastanGO-Notify sin tener que volver a probar o desplegar todo el Backend API. Esto reduce drásticamente el tiempo de "code freeze" y el riesgo de que un cambio en las notificaciones rompa el registro de transacciones.