Feedback Api

Arquitectura

Para poder favorecer la escalabilidad horizontal se optó por usar el patrón arquitectónico Rest, y se construyó una API con spring boot y java 8. Dicha API se encuentra distribuida en dos clusters:

- *feedback-job:* destinado a realizar tareas de reporte periódicas.
- *feedback-api:* destinado a atender consultas de clientes.

de esta forma se evita generar problemas de performance o disponibilidad que podrían perjudicar a los usuarios mientras se ejecutan queries de reporte.

Como storage principal se decidió utilizar una base relacional, específicamente *MySql*, dado que no considero que la estructura de los datos sean dinámicos.

Para mejorar los tiempos de procesamiento en las queries lanzadas a la base, se crearon los correspondientes índices que pueden observarse aquí. En caso de tener problemas de performance, se podría evaluar la necesidad de incorporar una caché distribuida[1] que decremente los tiempos de respuesta de la API, sin embargo en la medida de lo posible se optará por mantener una arquitectura sencilla.

Se utilizará el patrón publish/suscriber con el fin de comunicar novedades sobre todos los eventos de *feedbacks* a múltiples usuarios a la vez. Como tecnología para realizar esta tarea opte por el uso de RabbitMq, aunque también podría hacerse uso de Kafka.

Por último en cuanto a la estrategia utilizada para generar un reporte o ranking de stores basados en feedbacks de compradores, se decidió crear un status llamado *pending_report* que refiere a todos aquellos feedbacks que no fueron contabilizados para crear el ranking por tienda. De esta manera se disminuye el overhead de tener que traer todos los feedbacks de un store cada vez que se genera un ranking.

El reporte tendrá un resumen sobre los puntos acumulados por dicha tienda junto con la cantidad de feedbacks contabilizados, y en base a dicha información se podrá construir un score para dicho store. Los puntos acumulados por el reporte, serán modificados cada vez que se realice una actualización sobre el score de un feedback, o cada vez que el mismo sea eliminado.

[1] se dejó la dependencia de Memcached para ser usada en caso de ser necesaria

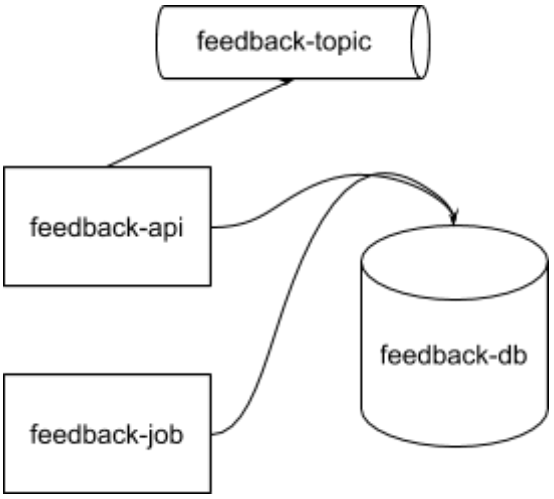

Figura 1: arquitectura propuesta para feedback-api

Documentación

endpoint	acción	descripción	body	response

/feedback	post	permite crear un nuevo feedback	{
			"order_id": 2,
			"seller_id": 1,
			"buyer_id": 1,
			"item_id": 1,
			"comment": "good",
			"store_id": "AR4",
			"score": "DIAMOND"
			}
				{
				"created_date": "2019-06-12T01:19:16.731+0000",
				"last_modified_date": "2019-06-12T01:19:16.731+0000",
				"order_id": 2,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,
				"comment": "este es otro comentario",
				"store_id": "AR4",
				"status": "PENDING_REPORT",
				"score": "DIAMOND",
				"reported": false
				}

/feedback/:id	get	obtener un feedback		{
				"created_date": "2019-06-12T01:19:17.000+0000",
				"last_modified_date": "2019-06-12T01:19:24.000+0000",
				"order_id": 2,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,
				"comment": "este es otro comentario",
				"store_id": "AR4",
				"status": "COMPLETED",
				"score": "DIAMOND",
				"reported": true
				}

	put	modificar un feedback	{
			"score": "BRONZE",
			"comment": "nice"
			}
				{
				"created_date": "2019-06-12T01:30:41.000+0000",
				"last_modified_date": "2019-06-12T01:30:55.209+0000",
				"order_id": 12,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,

				"comment": "nice", "store_id": "AR4", "status": "PENDING_REPORT", "score": "BRONZE", "reported": false }
	delete	eliminar un feedback		

/order/:id/feedback	get	obtener el feedback de una orden		{
				"created_date": "2019-06-12T01:19:17.000+0000",
				"last_modified_date": "2019-06-12T01:19:24.000+0000",
				"order_id": 2,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,
				"comment": "este es otro comentario",
				"store_id": "AR4",
				"status": "COMPLETED",
				"score": "DIAMOND",
				"reported": true
				}

/user/:id/feedbacks	get	listar todos los feedbacks de un comprador		[
				{
				"created_date": "2019-06-12T01:19:17.000+0000",
				"last_modified_date": "2019-06-12T01:19:24.000+0000",
				"order_id": 2,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,
				"comment": "este es otro comentario",
				"store_id": "AR4",
				"status": "COMPLETED",
				"score": "DIAMOND",
				"reported": true
				}
				]

/store/:id/feedbacks	get	listar todos los feedbacks para una tienda		[
				{
				"created_date": "2019-06-12T01:19:17.000+0000",
				"last_modified_date": "2019-06-12T01:19:24.000+0000",
				"order_id": 2,
				"seller_id": 1,
				"buyer_id": 1,
				"item_id": 1,
				"comment": "este es otro comentario",
				"store_id": "AR4",
				"status": "COMPLETED",
				"score": "DIAMOND",
				"reported": true
				}
				]

Tanto en los update (put) como en los create (post) se decidió devolver el la entidad actualizada o creada, para poder evitar potenciales lecturas posteriores.