



Universidad Nacional de Lanús

DESARROLLO DE SOFTWARE EN SISTEMAS DISTRIBUIDOS

Mensajería con Apache Kafka

Docentes:

- Ing Diego Andrés Azcurra
- Lic. Marcos Amaro

Integrantes:

- Lucas Nahuel Ambesi
- Dario Fabian Casais
- Daniel Alejandro Geier

Enunciado

Tomando como base el sistema creado en el trabajo práctico de RPC, desarrollar y modificar las funcionalidades detalladas en el apartado de requerimientos, utilizando Apache Kafka para producir y consumir los mensajes de los topics.

Requerimientos

- 1. Alta de receta:** cuando un usuario publica una receta, se deberá guardar en un topic de Kafka "Novedades", la siguiente información: nombre de usuario, título de la receta, url de la primera foto.
 - 2. Página de inicio:** al entrar al sistema, tiene que mostrarse una página de bienvenida que muestra las 5 últimas recetas creadas, de las cuáles se tomará la información consumiendo el topic de Kafka mencionado en el punto anterior. Esta página de inicio también contará con secciones para ver las recetas y usuarios más populares. Estos últimos no se consumen de ningún topic, sino de los datos que persista el consumidor de popularidad (punto 5).
 - 3. Usuarios:** en la funcionalidad para seguirlo, se guarda en el topic "PopularidadUsuario": nombreUsuario, puntaje (+1) si se comienza a seguir o (-1) si se deja de seguir.
 - 4. Recetas:** cuando un usuario entre a una receta, además de poder guardarla como favorita, también podrá comentarla y/o calificarla (de 1 a 5 estrellas).
 - a. Comentario:** esta acción no persiste en la base de datos directamente, sino que guardará en un topic "Comentarios" la información: usuarioComentario, recetaComentada, comentario. Los comentarios se guardarán en la base mediante un consumidor (punto 6).
- También guardará en un tópico "PopularidadReceta" de Kafka la siguiente información: id de la receta, puntaje (+1).
- b. Calificación:** también se guardará la información en el mismo topic, pero el puntaje corresponderá a las estrellas que el usuario le dio a la receta.

c. Favorita: guarda la información en el topic “PopularidadReceta”, pero puntaje (+1) si la marcó como favorita o con (-1) si la desmarcó. Además, aplica el mismo criterio pero guardando en el topic “PopularidadUsuario”: nombreUsuario del creador de la receta, puntaje (+-1) según el caso.

d. Visualización: agregar las modificaciones necesarias para mostrar el puntaje promedio de la receta y los comentarios de la misma.

e. Restricciones: el usuario que creó la receta, no puede calificar ni marcar como favorita su propia receta. Puede comentar, pero su comentario no cuenta para el cálculo de popularidad.

5. Popularidad: a. Receta: mediante un proceso programado que se ejecutará automáticamente cada un tiempo configurable (por ejemplo, cada 1 hora), se consumirá la información almacenada en el topic de “PopularidadReceta”, el cual hará el cálculo de popularidad con esos datos junto a los existentes en la base de datos. Luego de realizar los cálculos pertinentes, guardará esos resultados en la base de datos. b. Usuarios: igual al anterior, pero para el topic “PopularidadUsuario” y en un proceso aparte.

6. Persistencia de comentarios: similar al punto anterior, habrá un proceso programado que consuma los mensajes del topic “Comentarios” y guardará en forma masiva todos esos comentarios.

Resolución

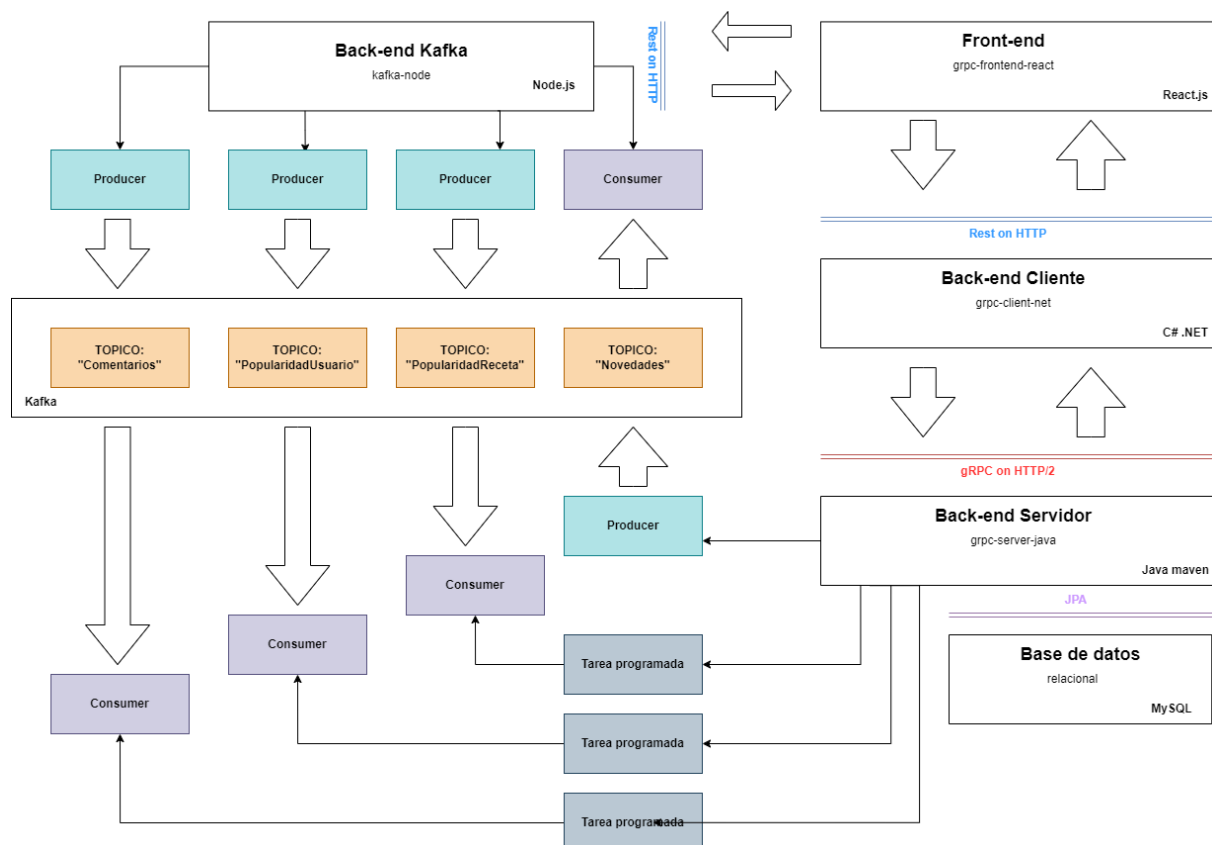
1. Estrategia de resolución

Para esta entrega y basándonos en los requisitos decidimos crear un nuevo proyecto llamado “kafka-node” en la tecnología Node.js para la manipulación de los producers y consumer que interactúan directamente con el frontend con la intención de abstraer estas nuevas funcionalidades y tener más control en su utilización.

Se puede hallar el código en el siguiente repositorio:

[lucasambesi/8627-1-TM-UNLa](https://github.com/lucasambesi/8627-1-TM-UNLa)

A continuación se mostrará la arquitectura de resolución planteada:



Ver en: [Arquitectura general.drawio](#)

1.1 Nuevas Recetas:

Producir en el servidor Java: Se creó un producer en el proyecto "grpc-java-server" utilizando la biblioteca kafka-clients. Este producer se encarga de enviar los datos de la receta (nombre de usuario, título y URL de la primera foto) al topic "Novedades" de Kafka.

Consumer en Node.js: Se desarrolló un consumer en el proyecto "kafka-node" usando la librería "kafkajs" para escuchar el topic "**Novedades**". Cuando recibe una petición HTTP al endpoint '/api/recipes/recipes' se activa el consumer con un nuevo grupo y de esta manera siempre se obtienen los últimos 5 mensajes leídos o no del tópico.

Visualización:

El endpoint '/api/recipes/recipes' es llamado por el frontend en la página de inicio y muestra las 5 últimas recetas creadas y se muestra en la vista:

CHEF EN CASA

EXPLORAR MIS RECETAS MI PERFIL CERRAR SESIÓN

Novedades



Receta de Masa para pizza a la piedra...

Creada por @lucas

[DETALLES](#)



Receta de Birriamen

Creada por @lucas

[DETALLES](#)



Receta de Birriamen

Creada por @lucas

[DETALLES](#)



Receta de Bizcocho de chocolate en fr...

Creada por @lucas

[DETALLES](#)



Receta de Tarta Massini clásica...

Creada por @lucas

[DETALLES](#)

Top usuarios

Lucas Ambesi
@lucas - Acumula una popularidad de 2 puntos.

Puesto #1

1.2 Popularidad de recetas:

Producir en Node.js: Se implementó un producer en el proyecto kafka-node para los topic "**PopularidadReceta**". Este es accedido mediante una callback que expone en el endpoint '/api/recipes/popularity'

Cálculo de Popularidad: Se creó un proceso programado en el proyecto "grpc-java-server" con la libreria "quartz" que dispara un consumer también creado previamente, este lee los mensajes de "**PopularidadReceta**" de Kafka. Estos datos se utilizan para calcular la popularidad de las recetas mediante el valor ya existente y los nuevos mensajes. Los resultados se almacenan en la base de datos MySQL en un nuevo campo creado en la tabla recipes llamado popularity

Persistencia de calificación por usuario:

Cuando el usuario selecciona una calificación es necesario guardar su valoración independientemente del impacto que esta tenga en la popularidad de la receta. Por lo que fue necesario crear una nueva tabla en la base de datos llamada Rating con los campos idRecipe, idUser y value.

Y se crearon nuevos endpoint de creación y modificación del rating, además de un nuevo archivo proto llamado Rating.

Se exponen del lado del cliente los siguiente endpoints para su acceso desde el frontend

RatingControllerClient		^
POST	/api/rating	▼
PUT	/api/rating	▼

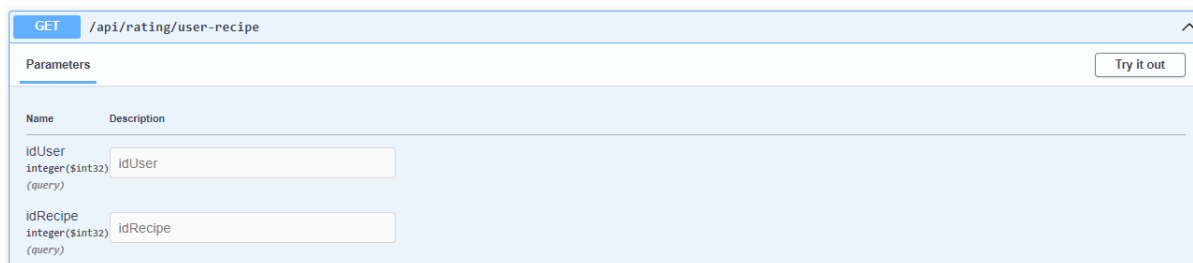
Visualización:

Para la selección de la valoración por usuario se creó un nuevo apartado en detalle de receta que le permita al usuario elegir de 1 a 5 estrella, al seleccionar las estrellas este dispara la persistencia en la base y el mensaje al tópico de kafka.

Tu valoración:

★ ★ ★ ☆ ☆ (3 estrellas)

Además fue necesario crear un nuevo endpoint a los ya mencionados para obtener esta información enviando un idUser y idRecipe, y es el siguiente:

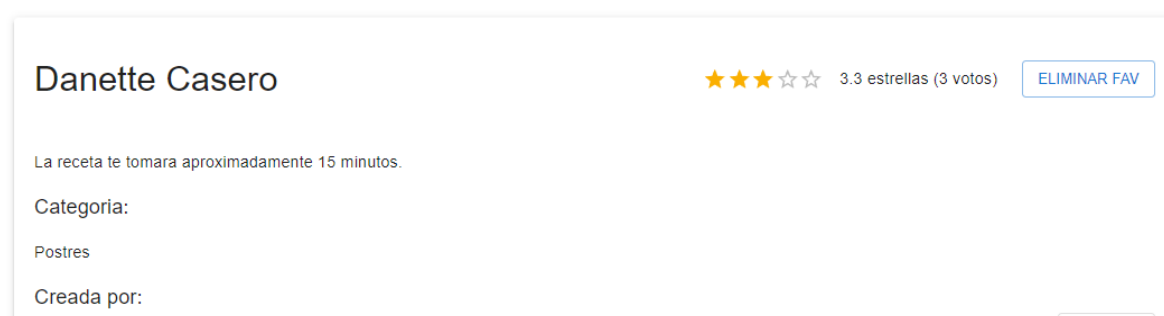


Swagger UI for the endpoint `GET /api/rating/user-recipe`. The interface shows a table with two parameters:

Name	Description
<code>idUser</code> <code>integer(\$int32)</code> <i>(query)</i>	<input type="text" value="idUser"/>
<code>idRecipe</code> <code>integer(\$int32)</code> <i>(query)</i>	<input type="text" value="idRecipe"/>

A "Try it out" button is located in the top right corner.

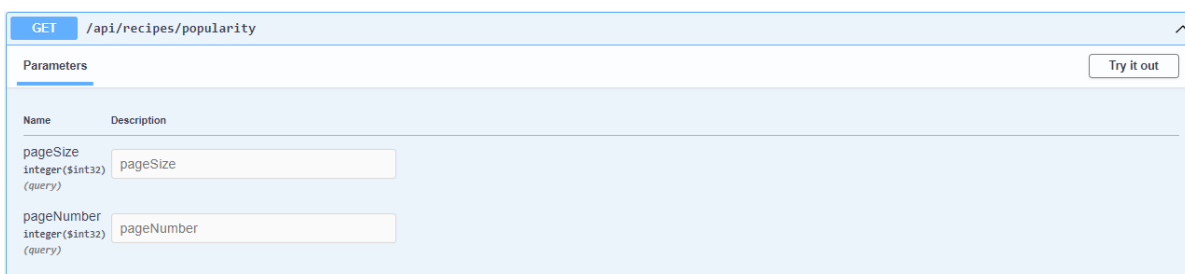
Para visualizar el promedio de valoración de una receta también fue necesario la creación de un nuevo endpoint donde le enviamos el id de una receta y este nos devuelve su average y la cantidad de valoraciones que obtuvo la receta. Este se llamada desde el frontend para ser visualizado en el detalle de la receta junto al título:



Recipe detail view for "Danette Casero". The interface includes:

- Recipe title: **Danette Casero**
- Rating: 3.3 estrellas (3 votos) with a star icon and an "ELIMINAR FAV" button.
- Preparation time: La receta te tomara aproximadamente 15 minutos.
- Category: Categoría:
- Postres
- Created by: Creada por:

Para visualizar el top de recetas por popularidad se creó un nuevo controlador en el backend expuesto por el cliente al frontend, que obtiene el top (a definir por parámetro) de las recetas ordenadas por popularidad.



Swagger UI for the endpoint `GET /api/recipes/popularity`. The interface shows a table with two parameters:

Name	Description
<code>pageSize</code> <code>integer(\$int32)</code> <i>(query)</i>	<input type="text" value="pageSize"/>
<code>pageNumber</code> <code>integer(\$int32)</code> <i>(query)</i>	<input type="text" value="pageNumber"/>

A "Try it out" button is located in the top right corner.

Desde la vista se consume el endpoint y se muestra en los detalles de la receta el siguiente apartado:

Top Recetas

Recena de Ensaladilla Vegana Acumula una popularidad de 6 puntos. La ensaladilla vegana consiste en un plato tradicional e ideal para el verano, es la tapa que siempre encuentras en cualquier bar.	Puesto #1 DETALLES
Receta de Espaguetti rojo sin crema Acumula una popularidad de 4 puntos. Aprende a preparar una de las pastas más populares con una salsa igualmente muy tradicional	Puesto #2 DETALLES
Emoliente Peruano Acumula una popularidad de 2 puntos. El emoliente es una bebida tradicional peruana elaborada con una base de hierbas, cereales, semillas y frutas.	Puesto #3 DETALLES
Receta de Masa para pizza a la piedra Acumula una popularidad de 0 puntos.	Puesto #4

1.3 Comentarios en recetas:

Producer en Node.js: Se implementó un producer en el proyecto kafka-node para los topic "**Comentarios**". Este es accedido mediante una callback que expone en el endpoint '/api/recipes/send-comment'.

Persistencia de comentarios: Se creó un proceso programado en el proyecto "grpc-java-server" que dispara un consumer también creado previamente, este lee los mensajes de "**Comentarios**" de Kafka. Estos datos se almacenan en la base de datos MySQL en una nueva tabla llamada "Comment" que posee los campos idComment, idRecipe, idUser y value.

Visualización:

Para visualizar los comentarios en cada receta se necesitó crear un nuevo controller que obtenga la lista de comentarios de manera página por receta. Dentro del backend se creó un nuevo archivo Proto "Comment" para la comunicación entre cliente y servidor. Y desde el cliente se expone el siguiente endpoint:

CommentControllerClient

GET /api/comments/recipe

Parameters

Try it out

Name	Description
IdRecipe integer(\$int32) (query)	IdRecipe
PageNumber integer(\$int32) (query)	PageNumber
PageSize integer(\$int32) (query)	PageSize

Responses

Code	Description	Links
200	Success	No links

Desde la vista se consume el endpoint y se muestra en los detalles de la receta el siguiente apartado:

Comentarios:

Enviar Comentario

ENVIAR

Lucas Ambesi
@lucas dijo : comentario para la receta kafka

Lucas Ambesi
@lucas dijo : otro comentario

Lucas Ambesi
@lucas dijo : otro mas

< 1 2 3 4 5 >

1.4 Popularidad de usuarios:

Producir en Node.js: Se implementó un producer en el proyecto kafka-node para los topic "**PopularidadUsuario**". Este es accedido mediante una callback que expone en el endpoint '/api/users/popularity'

Cálculo de Popularidad: Se creó un proceso programado en el proyecto "grpc-java-server" que dispara un consumer también creado previamente, este lee los mensajes de "**PopularidadUsuario**" de Kafka. Estos datos se utilizan para calcular la popularidad de los usuarios. Los resultados se almacenan en la base de datos MySQL en un nuevo campo creado en la tabla users llamado popularity

Página de Inicio: Para visualizar el top de usuarios se creó un nuevo controlador en el backend expuesto por el cliente al frontend, que obtiene el top (a definir por parámetro) de los usuarios ordenados por popularidad.

The screenshot shows a REST client interface for the endpoint `GET /api/user/popularity`. The **Parameters** tab is active, displaying two query parameters: `pageSize` (type `integer($int32)`) and `pageNumber` (type `integer($int32)`), both with input fields. A `Try it out` button is in the top right. Below, the **Responses** tab shows a table with one entry: status code `200`, description `Success`, and a `No links` status.

Name	Description
<code>pageSize</code> <small>integer(\$int32)</small> <small>(query)</small>	<input type="text" value="pageSize"/>
<code>pageNumber</code> <small>integer(\$int32)</small> <small>(query)</small>	<input type="text" value="pageNumber"/>

Code	Description	Links
200	Success	No links

Este endpoint es llamado por un nuevo apartado en el Home del frontend que visualiza la información.

Top usuarios		
Lucas Ambesi @lucas - Acumula una popularidad de 2 puntos.	Puesto #1	
diego perez @diego - Acumula una popularidad de 2 puntos.	Puesto #2	
Dario Cassais @dario - Acumula una popularidad de 1 puntos.	Puesto #3	

2. Kafka

Para instalar kafka decimos utilizar docker, una vez instalado docker mediante este “docker-compose.yml” instalamos Kafka.

<https://github.com/lucasambesi/8627-1-TM-UNLa/blob/main/TP2%20-%20Kafka/docker-compose.yml>

Una vez instalado Kafka utilizamos la herramienta “Offset Explorer” para crear los tópicos y visualizar los mensajes:

The screenshot shows the Apache Kafka Explorer 2.3 interface. The 'Partitions' tab is selected, displaying a table of 24 partitions for the topic 'chufensca-PopularidadUsuario'. The table columns are Partition, Offset #, Key, Value, Filter, Messages, and Timestamp. The partitions are numbered 0 to 23, and the values are represented as (offset, score) pairs. The timestamps range from 2023-10-02 19:14:38.660 to 2023-10-02 19:20:09.502.

Partition	Offset #	Key	Value	Filter	Messages	Timestamp
0	54		(10Recipe-1, score=-1)			2023-10-02 19:14:38.660
1	55		(10Recipe-4, score=-2)			2023-10-02 19:14:38.660
2	56		(10Recipe-10, score=-4)			2023-10-02 19:14:58.829
3	57		(10Recipe-15, score=-1)			2023-10-02 19:15:01.145
4	58		(10Recipe-3, score=-1)			2023-10-02 19:16:01.805
5	59		(10Recipe-3, score=-4)			2023-10-02 19:16:05.724
6	60		(10Recipe-6, score=-1)			2023-10-02 19:16:10.319
7	61		(10Recipe-9, score=-2)			2023-10-02 19:16:18.087
8	62		(10Recipe-2, score=-1)			2023-10-02 19:16:27.636
9	63		(10Recipe-2, score=-4)			2023-10-02 19:16:30.348
10	64		(10Recipe-21, score=-1)			2023-10-02 19:16:51.038
11	65		(10Recipe-17, score=-1)			2023-10-02 19:18:38.355
12	66		(10Recipe-17, score=-1)			2023-10-02 19:19:39.397
13	67		(10Recipe-23, score=-3)			2023-10-02 19:19:52.471
14	68		(10Recipe-23, score=-1)			2023-10-02 19:19:53.261
15	69		(10Recipe-24, score=-1)			2023-10-02 19:20:05.674
16	70		(10Recipe-24, score=-1)			2023-10-02 19:20:09.502
17	71		(10Recipe-10, score=-1)			2023-10-02 19:20:09.502

3. Pruebas y demostración

Para este punto decidimos grabar un video mostrando las nuevas funcionalidades. Compartimos el link:

<https://youtu.be/yY9kPoZBlf8>

4. Participación:

Para desarrollar estas funcionalidades decidimos dividir el trabajo entre todos los integrantes de la forma más equitativa según nuestro punto de vista, para luego unirlos todo y realizar las conexiones.