INF-343 SISTEMAS DISTRIBUIDOS

TAREA Nº2: "Uso de servicios de comunicación de middleware en una aplicación distribuida simple de comercio electrónico"

01 de octubre de 2022

ANTECEDENTES GENERALES

Objetivos:

- Aprender a programar una aplicación distribuida usando servicios de comunicación de middleware: invocación remota, sistema de colas de mensajes (MoM) y servicios web.

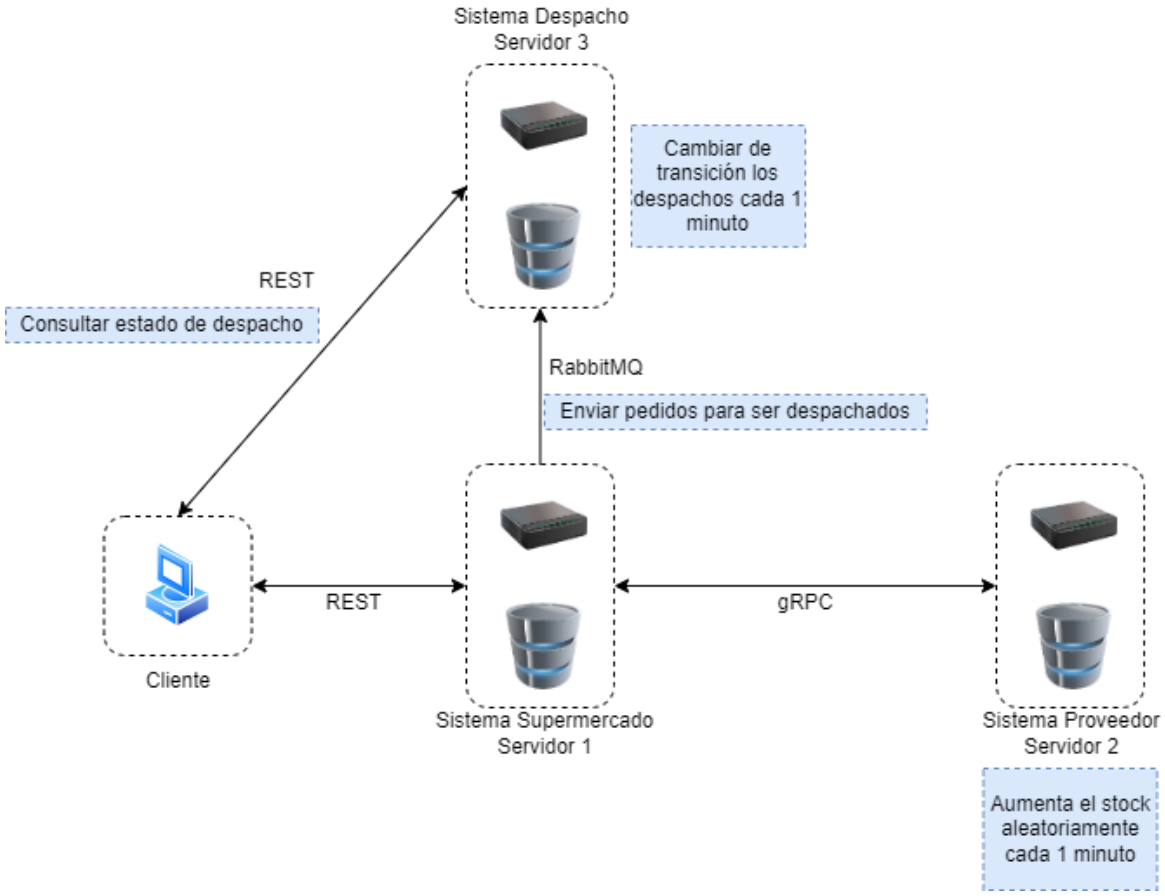
Fechas importantes:

Ayudantía por Zoom:	Jueves, 13 de octubre de 2022
Fecha de entrega de la tarea:	Viernes, 4 de noviembre de 2022

ESPECIFICACIÓNDE LA TAREA

Descripción de la aplicación

Considerando lo desarrollado en la tarea Nº1, en esta oportunidad se añadirán dos nuevos componentes. El primero, es un sistema que suministrará al supermercado ciertos productos cuando exista una carencia en el stock y necesidad de suministrarlos por un proveedor. El segundo, es un sistema que se encargará de despachar las compras una vez que el supermercado las notifique. Este último también da la posibilidad al cliente de consultar el estado de su despacho. Para un mejor entendimiento, observe el siguiente diagrama:



La comunicación entre el supermercado y el proveedor es a través de gRPC. En la tarea Nº1, cuando el supermercado se quedaba sin stock para un producto o si este no era suficiente para un pedido, no sucedía nada posteriormente, por lo que en esta oportunidad lo que se requiere es que el supermercado solicite la cantidad necesaria para reabastecer el producto.

El proveedor controla su propia base de datos, donde contiene una tabla con productos y la cantidad que dispone para cada uno de ellos. Debido a que este sistema no tiene un stock infinito, cada 1 minuto reabastece los productos que tienen una cantidad 0 (la nueva cantidad corresponderá a un número aleatorio entre 1 y 10). Si el supermercado solicita aumentar el stock de algún producto y el proveedor no lo tiene, el proveedor añadirá el producto a su base de datos con un stock igual al número de unidades requeridas más 5 unidades adicionales, luego entregará las unidades requeridas al supermercado.

Con respecto al sistema de despacho, este mantiene una comunicación a través del sistema de mensajería RabbitMQ para recibir los pedidos que son enviados desde el supermercado. Cada pedido que es recibido por el sistema de despacho es persistido en una base de datos con un atributo adicional llamado “estado”, el cual permite conocer la fase en la que se encuentra el despacho; estas fases son: “RECIBIDO”, “EN_TRANSITO” y “ENTREGADO”. Para simular las transiciones entre las fases, cada 1 minuto se realizará una actualización general

considerando los registros que están dentro de la base de datos. A modo de ejemplo, observe el siguiente diagrama:

ID_DESPACHO	ESTADO
1000000	RECIBIDO
1000002	EN_TRANSITO
1000003	RECIBIDO
1000004	ENTREGADO

00:00:00 24-09-2022

ID_DESPACHO	ESTADO
1000000	EN_TRANSITO
1000002	ENTREGADO
1000003	EN_TRANSITO
1000004	ENTREGADO

00:01:00 24-09-2022

ID_DESPACHO	ESTADO
1000000	ENTREGADO
1000002	ENTREGADO
1000003	ENTREGADO
1000004	ENTREGADO

00:02:00 24-09-2022

Otro atributo adicional que es persistido a la base de datos es el ID del despacho, el cual corresponde a un número aleatorio entre 10^6 y 9 x 10^6. Este número se genera en el supermercado.

Otra función que cumple el sistema de despacho es permitir al cliente consultar el estado de un despacho en particular. En este caso, la comunicación es a través de una API REST, donde la solicitud incluye el ID del despacho y la respuesta es el estado correspondiente.

Arquitectura:

La arquitectura considera 1 cliente y 3 servidores, cada uno se detalla a continuación:

1. Cliente: Desde aquí se ejecuta el menú visto en la tarea 1, considerando que ahora se debe añadir una opción para consultar el estado de un pedido. Este cliente se ejecutará en el servidor 1.
2. Servidor 1: Este servidor incluirá (además del Cliente) al sistema de supermercado realizado en la tarea 1, añadiendo las modificaciones respectivas para comunicarse con el sistema de proveedor y despacho.
3. Servidor 2: Este servidor incluirá el sistema de proveedor. Aquí debe ejecutarse el servicio de gRPC en un archivo llamado “server.py”. El servicio debe habilitarse para el puerto 9000. Cualquier archivo con extensión “.proto” se puede definir libremente. Por último, debe haber un script (scheduled task) que se ejecute cada 1 minuto para actualizar productos con un stock 0. Todo este sistema debe estar escrito en Python.
4. Servidor 3: Este servidor incluirá el sistema de despacho. Aquí se debe instalar el servicio de RabbitMQ. También, un archivo que reciba los mensajes y otro que inicialice un servicio API REST para las peticiones que realizará el cliente. El tipo de Exchange para el servicio de RabbitMQ debe ser “Directo”. Por último, debe haber

un script (scheduled task) que se ejecute cada 1 minuto para actualizar los estados de los despachos.

Especificación de las interfaces:

A continuación, se especifica el *endpoint* que permitirá al cliente comunicarse con el sistema de despacho para conocer el estado respectivo.

Endpoint	10.X.X.X/api/clientes/estado_despacho/:id
Method	GET

Response	{
	"id_despacho": 1000000,
	"id_compra": 2,
	"estado": "EN_TRANSITO"
	}

Ejemplos:

```
Bienvenido

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 1
Ingrese su id: 8080
Ingrese su contraseña: 8931441
Inicio de sesión exitoso

Opciones:
1. Ver lista de productos
2. Hacer compra
3. Consultar despacho
4. Salir
Ingrese una opción: 2
Ingrese cantidad de productos a comprar: 2
Ingrese producto 1 par id-cantidad: 14-2
Ingrese producto 2 par id-cantidad: 20-5
Gracias por su compra!
Cantidad de productos comprados: 7
Monto total de la compra: 750
El ID del despacho es: 1000000

Opciones:
1. Ver lista de productos
2. Hacer compra
3. Consultar despacho
4. Salir
Ingrese una opción: 3
Ingrese el ID del despacho: 1000000
El estado del despacho es: RECIBIDO
```

```
Opciones:
1. Ver lista de productos
2. Hacer compra
3. Consultar despacho
4. Salir
Ingrese una opción: 4

Opciones:
1. Iniciar sesión como cliente
2. Iniciar sesión como administrador
3. Salir
Ingrese una opción: 3

Hasta luego!
```

Consideraciones:

- **Sistema Supermercado:**
  - La API debe recibir peticiones a través del puerto 5000, mientras que la base de datos en el puerto 3306.
  - La base de datos debe llamarse “tarea_1_sd”.
  - El usuario que se conecta a la base de datos debe llamarse “admin” y la contraseña es “12345678”.
- **Sistema Proveedor:**
  - La base de datos debe llamarse “db_inventario”.
  - El usuario que se conecta a la base de datos debe llamarse “admin” y la contraseña es “12345678”.
  - El script para importar las tablas de está aquí.
- **Sistema de Despacho:**
  - La base de datos debe llamarse “db_despachos”.
  - El usuario que se conecta a la base de datos debe llamarse “admin” y la contraseña es “12345678”.
  - El script para importar las tablas de está aquí.
- **Otros:**
  - Para conocer los formatos de entrada y salida a través de la terminal, guíese rigurosamente por los ejemplos.
  - No es necesario realizar ninguna validación para los datos que se ingresan a través de la terminal, excepto los de inicio de sesión.
  - La versión de MySQL es la 8.0.X.
  - La versión de Go es la 1.19.X.
  - La versión de RabbitMQ es 3.11.X.
  - Para consultas, escriba a hugo.leyton@sansano.usm.cl.

Reglas de la tarea:

- La tarea se entrega en grupos de 3 personas.
- La fecha de entrega es el día 4 de noviembre a las 23:55.
- La tarea debe ser subida al mismo repositorio creado en la tarea nº1.
- Todo el contenido de la tarea debe estar en una nueva rama llamada “tarea_2”.
- En el README del repositorio, incluya el nombre y apellido de cada integrante.

RÚBRICA

Se considera evaluar los siguientes puntos para la tarea Nº2.

1. Desarrollo de servidores:
- Servidor de supermercado (lógica de negocio, sin detalles de comunicación).
- Servidor de Proveedor (lógica de negocio, sin detalles de comunicación).
- Servidor de Despacho (lógica de negocio, sin detalles de comunicación).
- Documentación y calidad del código.

2. Desarrollo de un cliente de comercio:
- Calidad de la interfaz de presentación para el usuario (con su extensión)
- Correcta invocación de los servicios de la API RESTful (supermercado+despacho)
- Documentación y calidad del código.

3. Uso de gRPC y ProtoBuffer (Supermercado: Go – Proveedor; Python):
- Configuración del *runtime* para gRPC.
- Especificación IDL para gRPC y mensajes con ProtoBuffer.
- Compilación de stubs.

4. Configuración del servicio RabbitMQ:
- Funcionamiento correcto del servicio.
- Configuración de cola de para conectar Supermercado-Despacho.

5. Programación de mensajería (MoM) usando de RabbitMQ:
- Programación del productor de mensajes (Supermercado).
- Programación del consumidor de mensajes (Despacho).