INF-343 SISTEMAS DISTRIBUIDOS

TAREA Nº2: "Uso de servicios de comunicación de Middleware en una aplicación distribuida simple de negocio"

05 de octubre de 2023

ANTECEDENTES GENERALES

Objetivos:

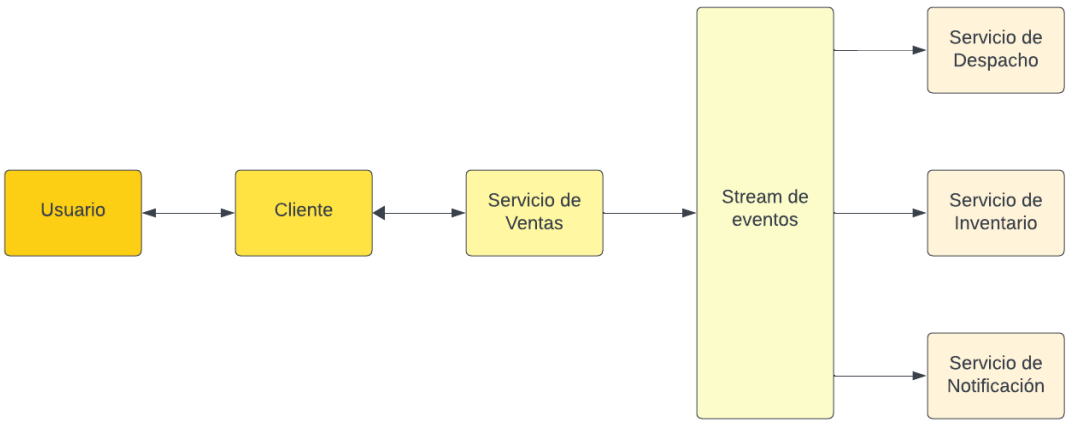
- Aprender a programar una aplicación distribuida usando servicios de comunicación de middleware: invocación remota, sistema de colas de mensajes (MoM) y servicios web.

Fechas importantes:

Ayudantía por Zoom: viernes 06 de octubre de 2023
Fecha de entrega de la tarea: miércoles 25 de octubre de 2023

ESPECIFICACIÓN DE LA APLICACIÓN

Una tienda de libros está creciendo constantemente, por lo que las visitas a su sitio web se han visto incrementadas y en algunos momentos del día la latencia es tan alta, que el sitio web no responde. La arquitectura actual está basada en contener todo en un único servicio para procesar las órdenes, los despachos y las notificaciones a los clientes, además de desplegar el sitio web. Aumentar los recursos de infraestructura no es una opción, ya que los costos serán elevados, por lo que se ha decidido en implementar una arquitectura basada en eventos, tal como lo muestra el siguiente diagrama:



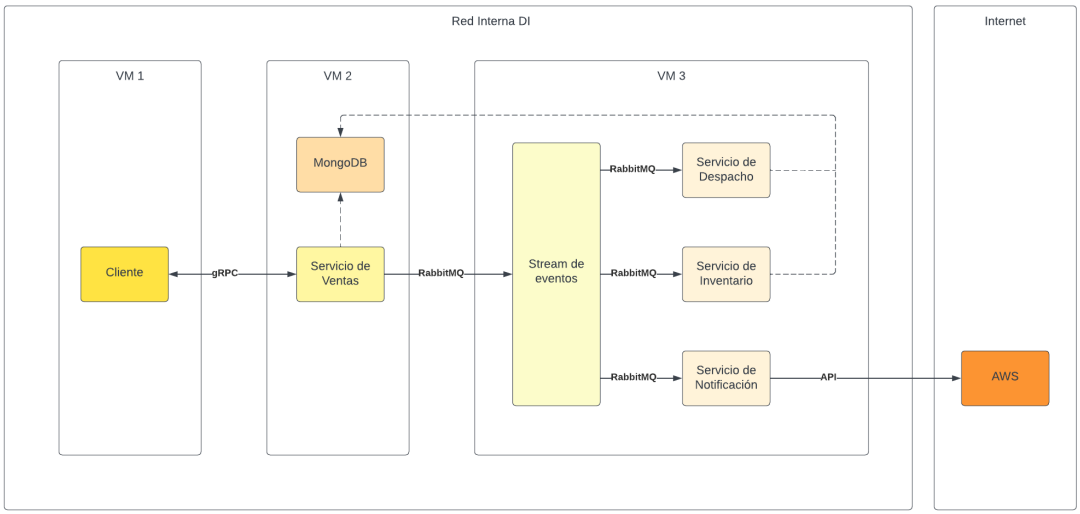
Cuando un usuario interactua con la plataforma a través de un cliente, selecciona los productos y posteriormente confirma su compra. Luego, el servicio de venta crea una orden y se comunica a través de eventos a los servicios de despacho, inventario y notificación. La tarea de cada uno de tales servicios se detalla a continuación:

1. **Servicio de Despacho:** Crea una orden de despacho que incluye un número de identificación único, datos del cliente y su ubicación.

2. **Servicio de Inventario:** Cada vez que una orden se crea, este servicio se encarga de actualizar el *stock* de cada uno de los productos implicados en la orden.
3. **Servicio de Notificación:** Este servicio se encarga de enviar un email al cliente para darle a conocer que su orden se está procesando.

DESARROLLO

Para esta tarea, se requiere implementar la siguiente arquitectura, utilizando API, RabbitMQ y gRPC. Para persistir los datos, se utilizará MongoDB.



Nótese que se empleará el uso de tres máquinas virtuales, las cuales ya fueron entregadas al comienzo del curso. A continuación, se describe el uso detallado de cada una de ellas:

1. **VM 1:** Esta máquina hospedará al cliente de la aplicación, un archivo llamado `client.go` que a través de su ejecutable recibirá como parámetro el nombre de un archivo `.json`, el cual contendrá todos los productos asociados a la compra. Una vez que se ejecuta el cliente, se comunicará a través de gRPC al servicio de ventas para entregarle los productos a considerar para la orden.
2. **VM 2:** Esta máquina contendrá un servicio de ventas para recibir una orden de compra y hacerla persistir en MongoDB en la colección `orders`. Una vez que esto esté finalizado, se retornará al cliente el `ID` de la orden creada. Así mismo, a través de un cliente de RabbitMQ, comunicará la estructura de la orden para que sea recibida por los servicios correspondientes. El servicio de ventas será inicializado a través de un archivo llamado `ventas.go`.
3. **VM 3:** Esta máquina hospedará el servicio de RabbitMQ para recibir los mensajes desde el servicio de ventas, también publicará los mensajes necesarios a los servicios de despacho, inventario y notificación. Cada uno de ellos se detalla a continuación:
    a. **Servicio de Despacho:** Una vez que recibe la estructura de la orden, adiciona a la orden persistida en MongoDB el detalle del despacho en el atributo `deliveries`.
    b. **Servicio de Inventario:** Una vez que recibe la estructura de la orden, por cada producto reduce el *stock* correspondiente en la colección `products` en MongoDB. Los archivos para escuchar los mensajes deben tener los siguientes nombres:

i. despacho.go
ii. inventario.go
iii. notificacion.go

c. **Servicio de Notificación:** Una vez que recibe la estructura de la orden, se comunica con una API de AWS para enviar un email al cliente correspondiente.

La estructura de una orden una vez creada y con un despacho añadido se detalla a continuación. Este registro está contenido en la colección orders:

```
{
  "orderID": "651e2da96d22f170036a919f",
  "products": [
    {
      "title": "The Lord of the Rings",
      "author": "J.R.R. Tolkien",
      "genre": "Fantasy",
      "pages": 1224,
      "publication": "1954",
      "quantity": 2,
      "price": 20.00
    }
  ],
  "customer": {
    "name": "John",
    "lastname": "Doe",
    "email": "juan.perez@example.com",
    "location": {
      "address1": "123 Main Street",
      "address2": "Apt. 1",
      "city": "Anytown",
      "state": "CA",
      "postalCode": "91234",
      "country": "USA"
    },
    "phone": "555-555-5555"
  },
  "deliveries": [
    {
      "shippingAddress": {
        "name": "John",
        "lastname": "Doe",
        "address1": "123 Main Street",
        "address2": "Apt. 1",
        "city": "Anytown",
        "state": "CA",
        "postalCode": "91234",
        "country": "USA",
        "phone": "555-555-5555"
      },
      "shippingMethod": "USPS",
      "trackingNumber": "12345678901234567890"
    }
  ]
}
```

Para el caso de los productos, se utiliza la siguiente estructura en la colección products:

```
{
  "title": "The Lord of the Rings",
  "author": "J.R.R. Tolkien",
  "genre": "Fantasy",
  "pages": 1224,
  "publication": "1954",
  "quantity": 98,
  "price": 20.00
}
```

Para comunicarse con la API de AWS para el envío de email, se considerará los siguientes datos:

Endpoint	Por definir
Método	POST

Payload	<pre>{
	  "orderID": "651e2da96d22f170036a919f",
	  "to": "jhon.doe@gmail.com",
	  "name": "Jhon",
	  "lastname": "Doe",
	  "products": [
	  {
	  "title": "The Lord of the Rings",
	  "author": "J.R.R. Tolkien",
	  "genre": "Fantasy",
	  "pages": 1224,
	  "publication": "1954",
	  "quantity": 2,
	  "price": 20.00
	  }
	  ]
	}</pre>

Respuesta (201)	<pre>{
	  "message": "Email sent successfully"
	}</pre>

Respuesta (4XX)	<pre>{
	  "message": "There was an error. The email was not sent."
	}</pre>

Respuesta (5XX)	<pre>{
	  "message": "There was an error. The email was not sent."
	}</pre>

REGLAS DE ENTREGA Y EVALUACIÓN

1. Solo se realizarán 5 ejecuciones para revisar la tarea:

Comando	**Descripción**
`./path_1/cliente products.json`	Ejecutar el cliente con los productos
`go run path_2/ventas.go`	Inicializa el servicio de ventas con gRPC.
`go run path_3/despacho.go`	Inicializa el consumidor de mensajes de RabbitMQ para el servicio de despacho.
`go run path_3/inventario.go`	Inicializa el consumidor de mensajes de RabbitMQ para el servicio de inventario.

`go run path_4/notificacion.go`	Inicializa el consumidor de mensajes de RabbitMQ para el servicio de notificaciones.

2. Todos los desarrollos tienen que ser subidos a sus respectivas máquinas virtuales.
3. Debe encargarse de instalar e inicializar los servicios de RabbitMQ y MongoDB, además de instalar todas los paquetes necesarios.
4. La tarea tiene que ser subida en un repositorio a https://gitlab.inf.utfsm.cl y añada el usuario “hleyton” con permisos de “maintainer”. Considere usar una nueva rama llamada “tarea_2”

Para dudas, escribir a hugo.leyton@usm.cl