Prueba técnica de conocimiento en Java y Postilion

Nombre: Maria Stephanie Katherine Yepez Mendoza

Problema:

Uno de nuestros clientes se está expandiendo y ahora está implementando un nuevo canal de atención para sus productos financieros, por lo tanto, está en el proceso de definición del protocolo de comunicación con los servicios de procesamiento transaccional. Dicho en otras palabras, esta definiendo la mensajería para el envío y recepción de la información transaccional.

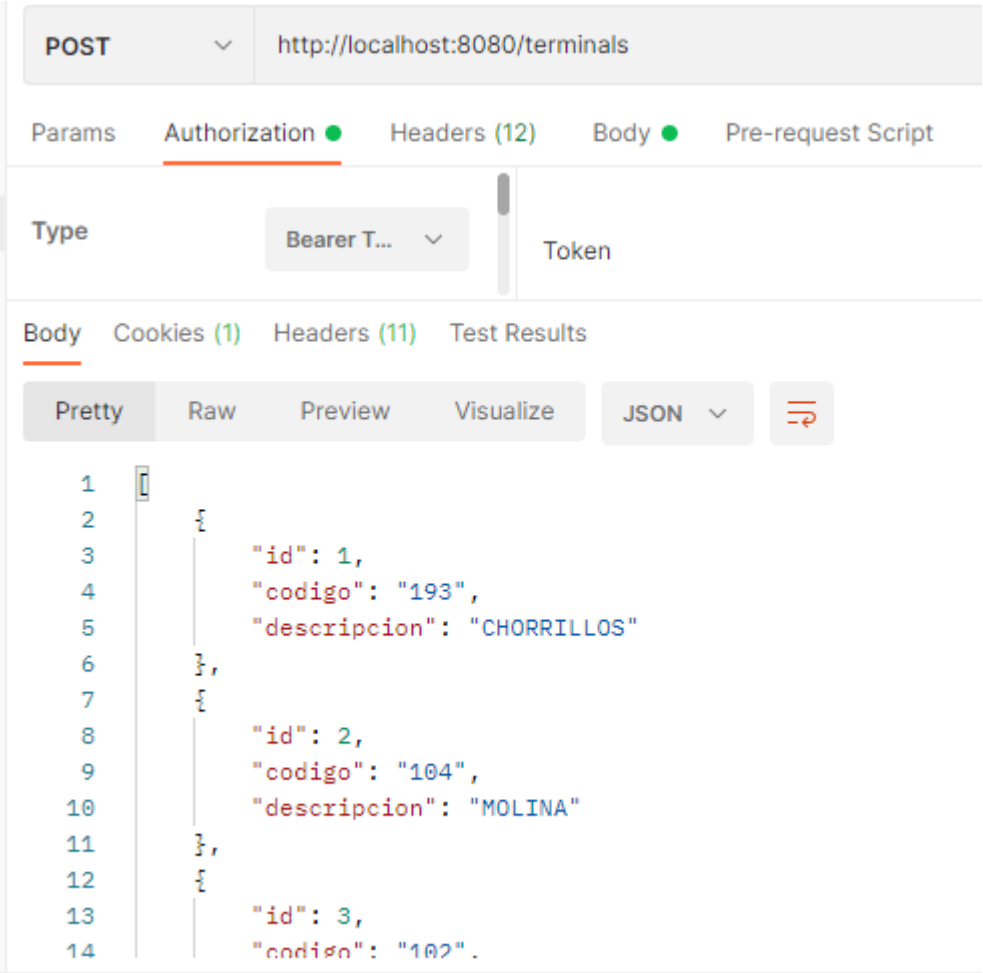
Las características de la mensajería más importantes que se han identificado hasta el momento son:

- Identificador para solicitud o respuesta.
- Identificador de la terminal que atendió.
- Fecha y hora de la transacción.
- Número de tarjeta o cuenta si está disponible.
- Al momento de enviar el mensaje, se debe imprimir como un texto plano y continuo.

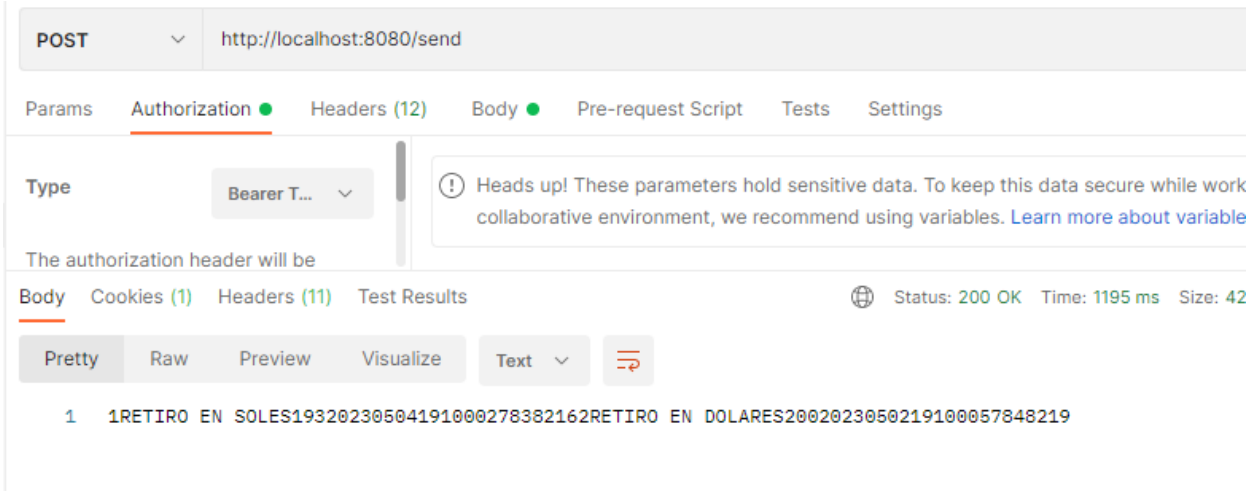
Si tiene más características que considere hacen falta, por favor inclúyalas en su modelo. Y no olvide incluir la documentación (JavaDoc) de todo su programa.

Se creó un microservicio (API Rest) en Spring Boot con diferentes endpoints para crear, modificar, eliminar y listar terminales, así como enviar/recibir mensajes con las características indicadas. Se trabajó con Hibernate, JPA, CRUD, lombok para crear métodos de atributos y JWT Token + Oauth2 (Seguridad). Se realizó pruebas en postman y la documentación (JavaDoc) se encuentra en la ruta **NovoPayment\backend\api\doc**

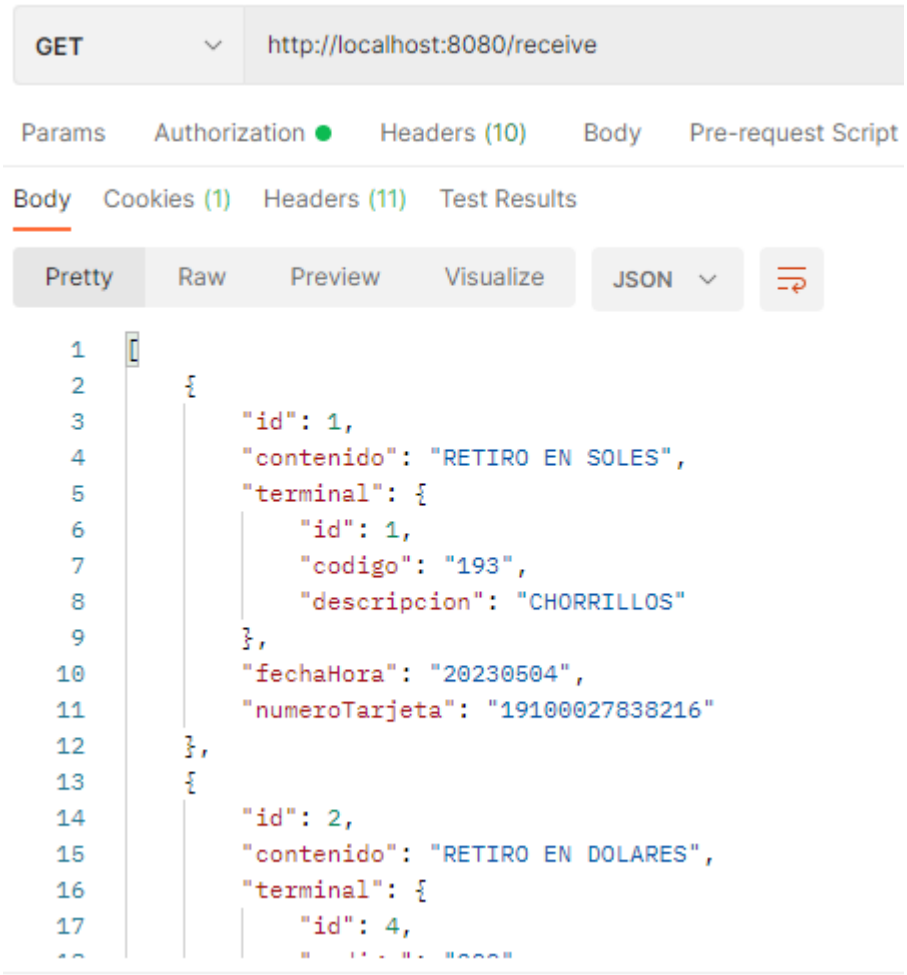
Crear terminales:



Enviar mensaje:



Recibir mensaje:



A continuación, encontrará un fragmento de código para que por favor lo analice detenidamente y responda las preguntas siguientes:

```
static PersistKey getInstace(String hsmIp, int hsmPort) {
    if (PersistKey.instace == null) {
        PersistKey.instace = new PersistKey(hsmIp, hsmPort);
    }
    return (PersistKey.instace);
}
```

1. ¿Qué tipo de objeto retorna el método *getInstance*?
   Retorna un objeto de tipo **PersistKey (clase).**

2. ¿Qué visibilidad tiene el método *getInstance*, y desde donde se puede invocar?
   Tiene una visibilidad de tipo **public** y se puede invocar desde cualquier lugar, siempre y cuando la clase tenga la misma visibilidad.

3. ¿Para invocar el método *getInstance* es necesario crear un objeto de la clase que lo contiene?
   No, ya que es un **método estático (static).**

4. El atributo *instance*, ¿es de clase o de instancia?
   Es de clase, ya que para acceder a él se utiliza la clase PersistKey.

5. ¿Qué tipo de dato tiene la variable *instance*?
   Es de tipo **PersistKey (clase).**

6. ¿Considera que se hace implementación de algún patrón de diseño? ¿Cuál?
   Si, el de **Singleton** ya que utiliza una condicional para validar si existe la instancia y la crea en caso de que no exista.

7. ¿Qué visibilidad debería tener el atributo *instance* asumiendo el patrón que indicó?
   Debe tener una visibilidad de tipo **private**.

8. ¿Qué visibilidad debería tener el constructor de la clase *PersistKey* asumiendo el patrón que indicó?
   Debe tener una visibilidad de tipo **private**.

Si y solo si ud. conoce de Postilion por favor mencione la consolas y productos que ha manejado e indique que es *tm_trans*, que entiende por Iso8583 Postilion e Integridad Transaccional.

ISO 8583 es un estándar internacional para el intercambio de mensajes electrónicos entre sistemas de pago y procesadores de transacciones financieras. Se utiliza para la comunicación entre los puntos de venta (POS) y los sistemas centrales de procesamiento de pagos.

La integridad transaccional se basa en el concepto ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), que es un conjunto de propiedades que se aseguran en una transacción. El objetivo es mantener la consistencia y la integridad de los datos antes y después de la transacción