Prueba tecnica – Backend

Implementación de un Sistema de Orquestación de Pagos – v1

Contexto:
En una plataforma de pagos para consumidores, los clientes pueden realizar pagos por diversos servicios o productos a través de un portal en línea. Para mejorar la experiencia de pago y la eficiencia operativa, la plataforma busca implementar un sistema backend que valide y orqueste las transacciones de pago. Este sistema garantizará un procesamiento fluido de los pagos.

Desafío:
Necesita implementar en Java Spring boot lo necesario para que maneje la orquestación para recibir pagos desde el frontend y comunicarse con otro servicio para validar y completar el pago. Es decir, se necesita un servicio que procese el pago de un cliente y se necesita un segundo servicio en el cual aplique el pago previo.

Tareas

Se te ha asignado la tarea de crear un sistema que:

1. **Diseño de Base de Datos:**
   a. Define un esquema relacional de base de datos para este sistema.
2. **Validar Entrada:**
   a. Asegura que el ID del cliente proporcionado y los detalles del pago sean válidos.
   b. Valida que el monto del pago cumpla con los requisitos, por ejemplo, que no sea negativo y esté dentro de los límites permitidos (definir un monto maximo permitido por transaccion)
3. **Procesar Pago:**
   a. Orquesta el pago enviando los detalles necesarios (ID del cliente, monto del pago, método de pago)
   b. Maneja cualquier error o caso excepcional, como detalles de pago inválidos o indisponibilidad del servicio.

   Una vez efectuado el pago, orqueste la transacción a través del servicio enviando el ID del cliente, el importe del pago y el método de pago, asegurándose de que el pago se procesa en una plataforma local (por ejemplo, un ERP o un sistema contable).

   La transaccion de pago recibida debe de ser persistida en la base de datos anteriormente diseñada.

4. **Generar Respuestas:**
   a. Devuelve una respuesta al frontend con el resultado de la operación, incluyendo mensajes de éxito o fallo.

Ejemplo de Uso

Escenario:

- Un cliente utiliza la plataforma para realizar un pago por un producto o servicio.
- Ejemplo de request:

```
{
  "customerId": "12345",
  "paymentAmount": 150.0,
  "paymentMethod": "credit_card"
}
```

Hint: ¿Consideras adicionar mas informacion al request?

Pasos para Implementar:

1. Validar el ID del cliente y los detalles del pago.
2. Asegurar que el monto del pago sea válido y cumpla con las reglas de la plataforma.
3. Orquestar el pago a través del servicio central enviando el ID del cliente, el monto del pago y el método de pago.
4. Manejar cualquier error devuelto por el servicio central (por ejemplo, fondos insuficientes, método de pago inválido).

Ejemplos de Response:

- **Exitoso:**

```
{
  "status": "success",
  "message": "Pago procesado con éxito."
}
```

- Fallido:

{

"status": "error",

"message": "El pago falló debido a un método de pago inválido."

}

Hint: ¿Consideras enriquecer esos responses?

Preguntas de Seguimiento

1. ¿Cómo diseñarías las pruebas unitarias para esta implementación para asegurar que se cubran todos los casos extremos?

   Crear pruebas unitarias para validaciones de negocio, estados límite (monto cero, cliente nulo, datos invalidos), respuestas esperadas y excepciones controladas.

2. ¿Qué cambios recomendarías en una revisión de código para que esta solución esté lista para producción?

   Manejo de errores correcto para no mostrar mensajes que no deberían mostrársele al usuario.

3. ¿Cómo asegurarías que el sistema sea resiliente a fallos en el servicio central de pagos?

   Manejar los casos de errores de manera estandarizada, en las replicaciones a base de datos utilizar notaciones para realizar rollback si fuera necesario, y utilizar auditoria en todas las tablas.

4. ¿Qué debería hacer la API que aplica el pago cuando recibe un error HTTP 500?

   Primero lo registraría en la base de datos como error, y luego dependiendo si se ha configurado que se realice mas intentos realizarlos y si el incidente continuo ya debe estar configurado un estado al cual pasará mi transacción para notificar a otra área encargada o en todo caso al usuario que no fue posible aplicar el pago.

5. ¿Cuál debería ser el comportamiento de la API cuando falla al procesar el pago?

   Ya debe estar configurado los casos de errores y seguimiento en los procesos que se realizaran por lo tanto debe comportarse de manera resiliente para manejar esos casos.

6. ¿Cómo debería manejar el sistema los reintentos si el servicio central no está disponible temporalmente?

   Se debe configurar la cantidad máxima de intentos a realizar caso contrario pasar a otra área específica u otro sistema que quizá maneje ese proceso con un tiempo de espera mayor entre reintentos quizá 2 horas o mas y si no notificar a un área encargada para proceder con la notificación al usuario por parte del orquestador del pago el marcara la transacción como FAIL.

Hint: Comprendemos que las herramientas de IA son tentadoras, pero en esta ocasión, sería ideal que nos sorprendas con tu propio razonamiento.

Entregables: Proyecto en gitX (github, gitlab, etc) + su respectiva documentacion (Documentación de código fuente y/o documentación de API).

Opcional: Servicios desplegados en cloud + collection de uso + cualquier entregable que consideres complementario.

Analizar las siguientes lineas en SQL

```
SELECT

case

  when wseManual.product_type_key = '6781f9a2-7237-446a-9266-3c1566ce73f6' then
'Merchandise'

   when wseManual.product_type_key = '745c8bef-8ae2-481f-af23-7303df3d22a6' then
'Cash'

  else

     'not running'

   end as product ,


count(*) quantity
```

```
  FROM user_request  ua
   LEFT JOIN credit_request cr ON ua.user_id = cr.id
   LEFT JOIN request_trn we ON ua.request_id = we.id
   LEFT JOIN request_trn_customer wec ON we.id = wec.id
   LEFT JOIN sales_evaluation wseManual ON ua.sales_id = wseManual.id
   LEFT JOIN evaluation_smart_credit wse_sm ON wseManual.id =   wse_sm.evaluation_id
   LEFT JOIN request_trn wsales ON wsales.customer_id = wseManual.customer_id
   LEFT JOIN request_trn_customer customer_sale ON customer_sale.workflow_emma_id
= wsales.id
        where
         (wseManual.created) BETWEEN '2022-08-01 00:00:00' and '2022-08-14 00:00:00'
group by wseManual.product_type_key;
```

Esta consulta busca contar registros agrupados por productos en tres categorías, “Merchandise, Cash, Not Running” y hace una seria de verificaciones adicionales haciendo Left Join, limita la búsqueda a un rango de fechas.

EOF