PRUEBA DE DESARROLLO BACKEND

credibanco

Introducción

Bank Inc es una empresa que permite asignar a sus clientes una tarjeta débito o crédito para realizar compras en los comercios asociados. Dicha tarjeta cuenta con las siguientes características:

- Tiene una longitud de 16 dígitos, donde los 6 primeros dígitos corresponden al id del producto y los demás son aleatorios.
- Nombre del titular de la cuenta (Primer nombre y apellido)
- Fecha de vencimiento (mm/aaaa) de 3 años posterior a la fecha de creación.
- Permite únicamente movimiento en dólares.

Cada vez que se requiera una nueva tarjeta el banco debe realizar un proceso de emisión de tarjeta, el cual consta de los siguientes pasos:

1. A partir del identificador de producto, generar el número de tarjeta.
2. Se debe realizar la activación de la tarjeta, ya que por controles de seguridad estan inactivas.
3. El saldo de dicha tarjeta es de cero pesos, por lo tanto, debe poderse hacer una recarga de saldo.

Una vez Bank Inc haga su proceso de emisión, la tarjeta estará disponible para realizar compras teniendo en cuenta que:

- Se debe contar con el saldo suficiente para realizar la compra (el saldo disponible no debe ser menos a cero).
- Debe estar vigente, es decir, la fecha de la transacción no debe ser mayor a la fecha de vencimiento.
- La tarjeta debió ser activada en el proceso de emisión
- La tarjeta no debe estar bloqueada, este es un proceso que realizan los administradores del sistema cuando encuentran alguna inconsistencia.

Al realizar una transacción de compra, el sistema asignará un identificador con el cual es posible consultar la transacción o realizar la anulación.

Requerimiento

Nivel 1

Bank Inc lo ha contratado para crear un sistema que permita administrar sus tarjetas y transacciones. Para esto, debe construir un API REST para:

1. Generar número de tarjeta

   Tipo de método: **GET**

   Recurso: **/card/{productId}/number**

2. Activar tarjeta

Tipo de método: POST

Recurso: **/card/enroll**

```
{
    "cardId": "1020301234567801"
}
```

3. Bloquear tarjeta

Tipo de método: DELETE

Recurso: **/card/{cardId}**

4. Recargar saldo

Tipo de método: **POST**

Recurso: **/card/balance**

```
{
    "cardId": "1020301234567801",
    "balance": "10000"
}
```

5. Consulta de saldo

Tipo de método: **GET**

Recurso: **/card/balance/{cardId}**

6. Transacción de compra

Tipo de método: **POST**

Recurso: **/transaction/purchase**

```
{
    "cardId": "1020301234567801",
    "price": 100
}
```

7. Consultar transacción

Tipo de método: **GET**

Recurso: **/transaction/{transactionId}**

Nivel 2

1. Crear el servicio de anulación de transacciones, el cual consiste en que a partir de la transacción de compra:
   - Se debe anular a partir del id de transacción
   - La transacción a anular no debe ser mayor a 24 horas.
   - La transacción quede marcada en anulada.
   - El valor de la compra debe volver a estar disponible en el saldo.

Tipo de método: **POST**

Recurso: **/transaction/anulation**

```
{
    "cardId": "1020301234567801",
    "transactionId": "102030"
}
```

2. Pruebas unitarias con una cobertura igual o mayor al 80%

Nivel 3

Desplegar el API en un servidor cloud (AWS, Google Cloud Platform, entre otros)

Entregables

- El código fuente debe estar en un repositorio privado (GitHub, GitLab, BitBucket, otros).
- Colección de postman, soap UI que permita probar la solución.
- Instrucciones para ejecutar la aplicación, local o cloud (si aplica)
- Documentación que permita el entendimiento de la solución. Ejemplo: flujo de datos, modelo de base de datos, manual de integración, swagger, listado de requerimientos obtenidos.
- Buenas prácticas de desarrollo. Tener en cuenta el uso de una buena arquitectura, manejo de excepciones, validación de los datos de entrada.

Consideraciones

- Usar spring framework para su desarrollo
- La base de datos debe ser relacional