Prueba técnica

Backend Java 17 + Spring Boot

Alcance

EndPoints: Emisión, activación, bloquea o consulta.

Nota: La tarjeta debe quedar ACTIVA.

#	Descripción
Tecnología	**Java 17, Spring Boot, Maven**
Persistencia	Base de datos relacional (PostgreSQL Recomendad)
Entrega	Repositorio Privado (Git)
Tiempo Estimado	1:30H – 2:00H máximo

Reglas y restricciones

Durante el desarrollo de la prueba **no está permitido**:

- Usar herramientas de IA.
- Consultar internet (búsquedas, foros, documentación, etc.).
- Reutilizar/Revisar código de otros proyectos (Personales o de terceros).

Contexto

Se requiere construir un API REST para la administración de tarjetas asociadas a productos. La prueba evalúa la calidad de código, diseño de API, validaciones y buenas prácticas.

Requisitos Funcionales

Emisión de tarjeta	Crear una tarjeta a partir de un producto y datos del titular. Generar cardNumber de 16 dígitos (Primeros 6 dígitos = Product Id)
Activación	Permitir activar una tarjeta emitida.
Bloqueo	Permitir bloquear una tarjeta por inconsistencia o seguridad.
Consulta	Consulta el detalle de una tarjeta (Estado, vencimiento, saldo).

Modelo de dominio mínimo

Se sugiere implementar como mínimo las siguientes entidades/campos:

Product: productId (6 digitos), name, type (DEBIT, CREDIT)

Card: cardId, cardNumber, holderName, issuedAt (fecha Emision), expiresAt (fecha Expliracion -> expiresAt + 3 years), status(ISSUED, ACTIVE, BLOCKED), blockedAt (fecha de bloqueo si está bloqueada, sino vacío), blockedReason(Razón del bloqueo sino vacío), balance (inicial 0.00), currency (USD).

Reglas generales

- productId debe ser exactamente de 6 dígitos numéricos.
- cardNumber debe ser único (contraint único en DB),
- Activar solo si status = ISSUED, Si ya está activa, la operación puede ser idempotente.
- Bloquear cambia status a BLOKED, Si ya esta BLOCKED, puede ser idempotente.
- Moneda Única: USD. Saldo inicial: 0.00.

API – EndPoints

1. **Emitir tarjeta**

   POST /cards

   Crea una tarjeta (Estado inicial ISSUED) y genera su número de 16 dígitos.

Request

```
{
"productId": "123456",
"holderName": "Ricardo Perez"
}
```

Response

```
{
"cardId": "uuid-o-id",
"cardNumber": "1234567891234567",
"status": "ISSUED",
"expiresAt": "2027-01-18",
"balance": 0.00,
"currency": "USD"
}
```

Reglas

- Los primeros 6 dígitos del cardNumber corresponden a productId
- Los 10 dígitos restantes son aleatorios.
- expiresAt = issuedAt + 3 years.

Status Codes de la respuesta esperados (Se recomienda)

- 201 Created
- 400 Bad Request
- 404 Not Fount (productId inexistente)

2. **Activar tarjeta**

POST /cards/{cardId}/activate

Activa una tarjeta previamente emitida. Una tarjeta ACTIVE queda conceptualmente habilitada.

Response

```
{
“cardId”: “uuid-o-id”,
“Status”: “ACTIVE”
}
```

Reglas

- Solo se pude activar si status = ISSUED.
- Si ya esta ACTIVE, se recomienda comportamiento idempotente (status response 200 - OK).
- Si esta BLOKED, no pude activarse.

Status codes de la respuesta esperados (Se recomienda)

- 200 OK
- 404 Not Found
- 409 Conflict (status invalido a asignar a la tarjeta)

3. **Bloquear tarjeta**

POST /cards/{cardId}/block

Bloquea una tarjeta por razones de seguridad o inconsistencia.

Request

```
{
“reason”: “SECURITY_INCONSISTENCY”,
}
```

Response

Reglas

- Si ya esta BLOCKED, puede ser idempotente.

```
{
“cardId”: “uuid-o-id”,
“status”: “BLOCKED”
}
```

- Al bloquear, registrar blockedAt y reason (Si aplica).

Status codes de la respuesta esperados (Se recomienda)

- 200 OK
- 404 Not Found

4. **Consultar tarjeta**

GET /cards/{cardId}

Obtiene el detalle de una tarjeta.

Response

```
{
“cardId”: “uuid-o-id”,
“cardNumber”: “1234567891234567”,
“holderName”: “Ricardo Perez”,
“status”: “ACTIVE”,
“blockedAt”: “”,
“blockedReason”: “”,
“expiresAt”: ”2027-01-18”,
“balance”:  0.00,
“currency”: “USD”
}
```

Status codes esperados (Se recomiendan)

- 200 OK
- 404 Not Found

Criterios de evaluación

- Diseño de API REST (status codes, endpoints).
- Manejo de base de datos relacional (Persistencia en base de datos).
- Reglas de negocio.

Entrega de solución

El candidato debe hacer la entrega de la solución en un repositorio privado (Git):

- Código fuente
- Base de datos.
- Instrucciones para ejecutar la aplicación.
- Ejemplos de uso (Curl o Postman) para emitir, consultar, activar y bloquear tarjeta.