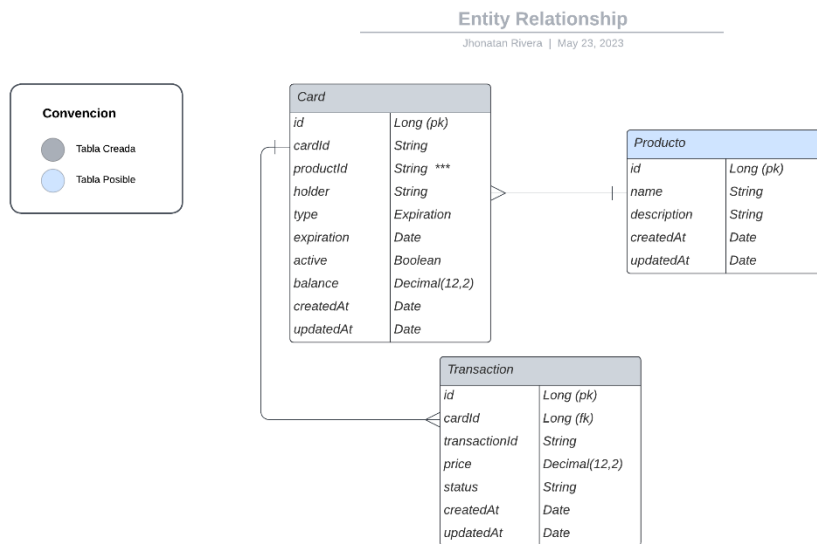


Modelo ER



Particularidades:

Para la realización del test evalué como posibles 3 entidades, que darían vida a 3 tablas en la DB, sin embargo, para efectos prácticos solo implementé 2 de ellas, Card y Transaction, por lo que la entidad Producto solo existe en este modelo pero se entiende que debería existir una tabla Producto relacionada con Card que describe que para cada Producto podría tener 1-N tarjetas asociadas.

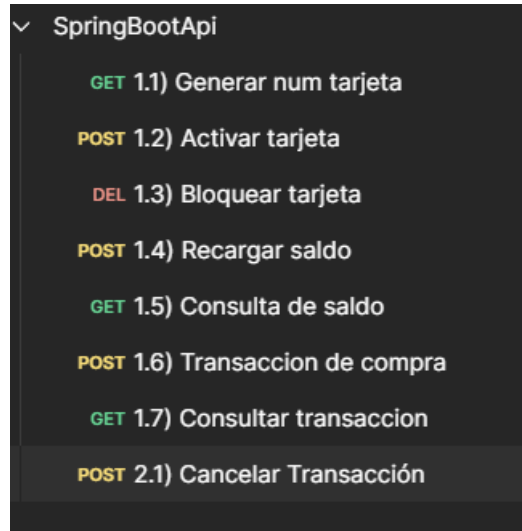
*** Por lo mencionado la tabla Card sigue teniendo un productId, y aunque este valor debería ser de tipo Long al venir de Producto, para simplificar el desarrollo de la prueba será de tipo String.

API

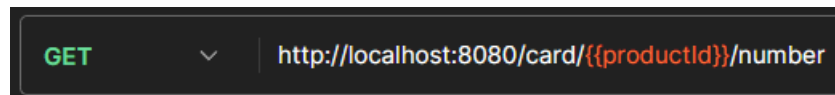
El API permite todos los tipos de consultas mencionados en la prueba, una posibilidad sería usar la colección de Postman que se provee en el mismo repositorio, es importante mencionar que si bien el puerto de la DB es dinámico y se puede cambiar tan solo cambiando la variable "DBPORT" del archivo .env (archivo que se debería crear después del clonado del repositorio), no sucede lo mismo, con el puerto del server, que por defecto será 8080.

Servicios

A continuación, se muestran los servicios junto con el Método HTTP, respetando y siguiendo **TODO**s los lineamientos de la descripción de la prueba. Posteriormente se encuentran uno a uno relacionados a su funcionamiento.

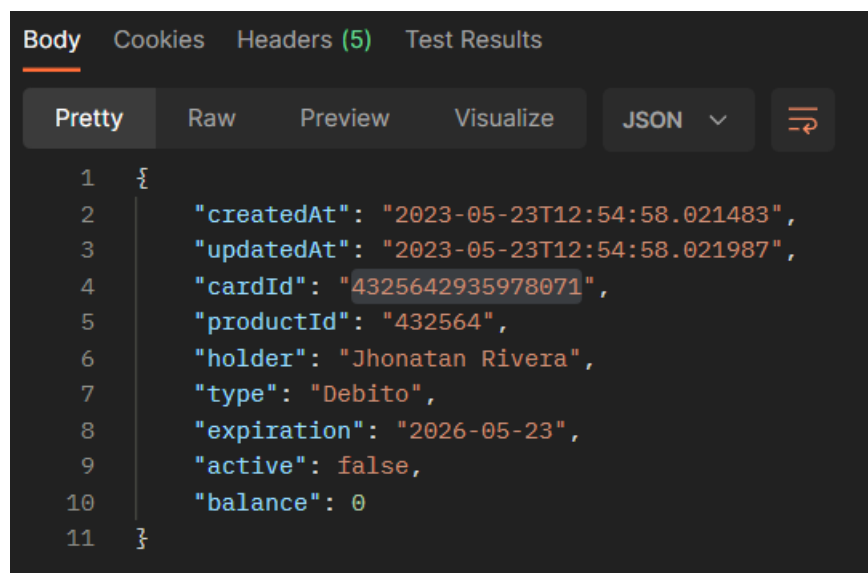


1.1) Generar número de tarjeta.



¿Qué hace?: Crea una tarjeta nueva, dependiendo del id del producto proporcionado.

Respuesta esperada: Objeto con los atributos de la tarjeta.



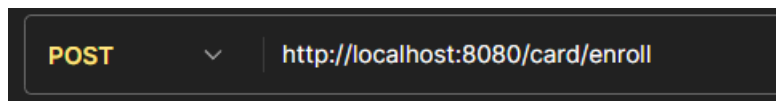
Excepciones:

- 1) Si el número del producto no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 6 dígitos numéricos.
- 2) Si el producto generado es duplicado (Caso particularmente poco probable): Retorna un mensaje de error genérico.

Consideraciones:

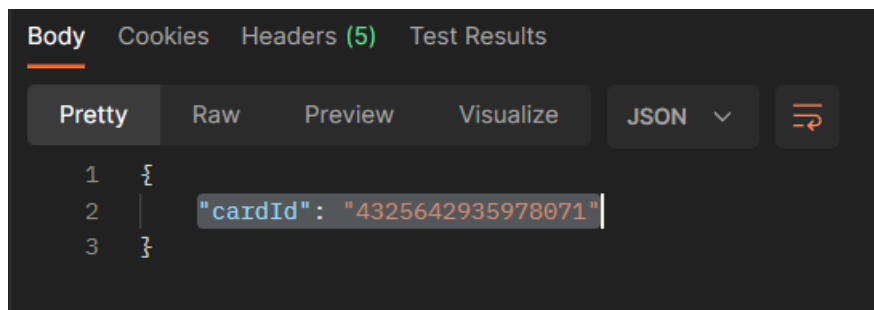
- Ya que el método usado es GET y para seguir los lineamientos de la prueba no añadido más información o lo cambio a un método más acorde como POST, los datos generados para atributos como "holder", y "type" son hardcodeados o estáticos. Un atributo como "holder" revela la relación que existen entre una entidad posible como Cliente o Usuario y tarjeta.
- La implementación de 6 dígitos de producto y 10 dígitos de la tarjeta de limita la cantidad de tarjetas generadas, y por lo tanto los 10 dígitos generados de manera aleatoria podrían dar lugar a colisiones, si bien la posibilidad al principio es bastante (y suficientemente) menor, en la medida que escale la cantidad de tarjetas por servicio esta empieza a crecer, de cualquier forma si se llegara el hipotético caso de colisión no se podría guardar debido a la naturaleza del campo cardId que es Unique.

1.2) Activar tarjeta



¿Qué hace?: Activa la tarjeta, dependiendo del id de tarjeta proporcionado en el cuerpo de la petición.

Respuesta esperada: Devuelve el id de la tarjeta en un objeto



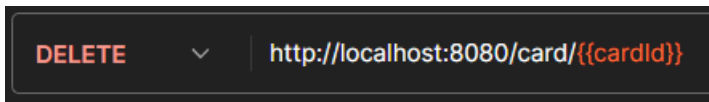
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.

Consideraciones:

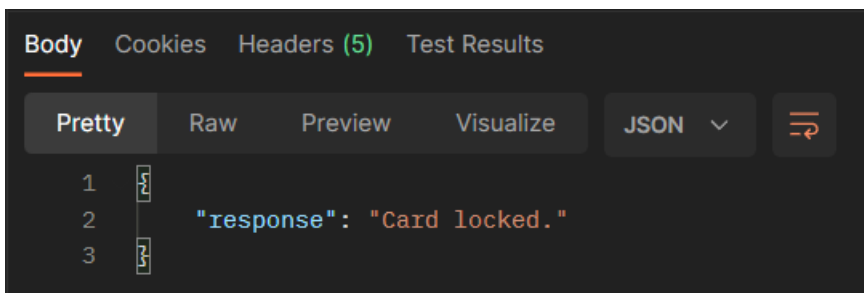
- Lo que hace internamente es solo cambiar el atributo Active a verdadero, si bien uno podría enviar un mensaje de activación previamente realizada en el caso de que este ya sea verdadero, y así evitar hacer peticiones de escritura innecesarias a la DB, lo que hace ahora es que sigue cambiándolo.

1.3) Bloquear tarjeta



¿Qué hace?: Bloquea la tarjeta, dependiendo del id de tarjeta proporcionado en el cuerpo de la petición.

Respuesta esperada: Devuelve el mensaje Tarjeta bloqueada.



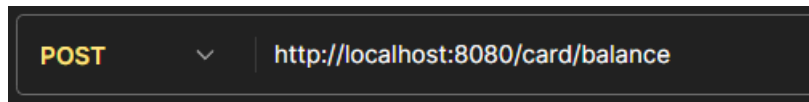
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.

Consideraciones:

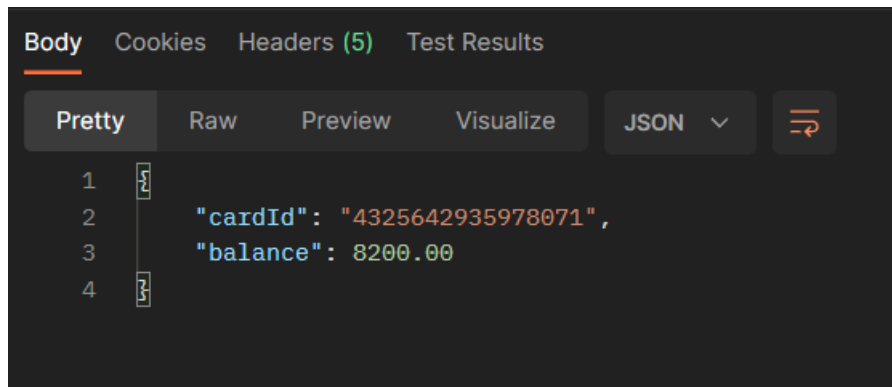
- Lo que hace internamente es eliminar la tarjeta, todo esto dentro del contexto de la prueba y para respetar la firma de la petición (delete sobre un recurso), sin embargo, en la vida real habría que considerar varios factores, y muy probablemente no debería eliminarse el registro, sino conservarse como un status de la tarjeta todo esto con el fin de conservar información que pudiese ser necesaria más adelante.
- Uno de los factores importantes es que pasa cuando la tarjeta todavía tiene saldo, esa información no se debería perder nunca.

1.4) Recargar saldo



¿Qué hace?: Añade saldo a la tarjeta, dependiendo del id de la tarjeta y el atributo balance, que vienen en el cuerpo de la petición.

Respuesta esperada: Devuelve un objeto con el cardId y el nuevo balance o saldo.



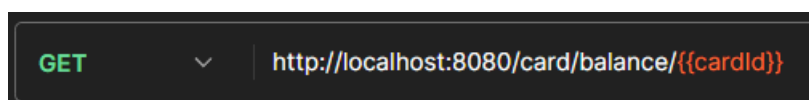
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.
- 3) Si el balance añadir NO es positivo: Retorna un mensaje diciéndolo.
- 4) Si la tarjeta se encuentra inactiva: Retorna un mensaje de tarjeta inactiva.

Consideraciones:

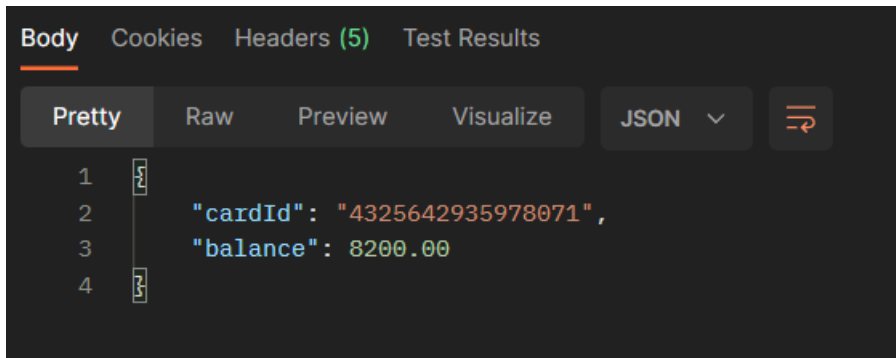
- Añadir balance suena como algo extraño, por el hecho de que estas operaciones deberían dejar una huella que permita la trazabilidad, por lo que quizá debiese existir una relación entre Transaction y este tipo de operación, al fin y al cabo, el dinero sale de algún lado y pasa a otro. En este momento sin embargo no se crea ningún registro en Transaction sino que solo lo añade si pasa las validaciones

1.5) Consultar saldo



¿Qué hace?: Consulta el saldo de la tarjeta, dependiendo del id de la tarjeta que viene en el cuerpo de la petición.

Respuesta esperada: Devuelve un objeto con el cardId y el saldo actual.



```
{
  "cardId": "4325642935978071",
  "balance": 8200.00
}
```

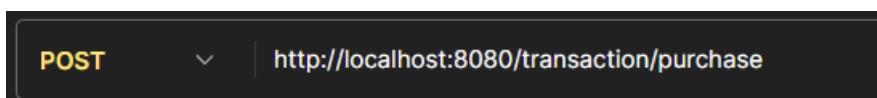
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.
- 3) Si la tarjeta se encuentra inactiva: Retorna un mensaje de tarjeta inactiva.

Consideraciones:

- Si bien en la implementación proporcionada el balance solo se añade, probablemente se debería tener en cuenta el tipo de tarjeta si es Débito o Crédito, debido a la manera en la que funcionan estas tarjetas, por ejemplo, el crédito es un cupo específico, en cambio en débito si se podría recargar en teoría “cuanto se desee”.

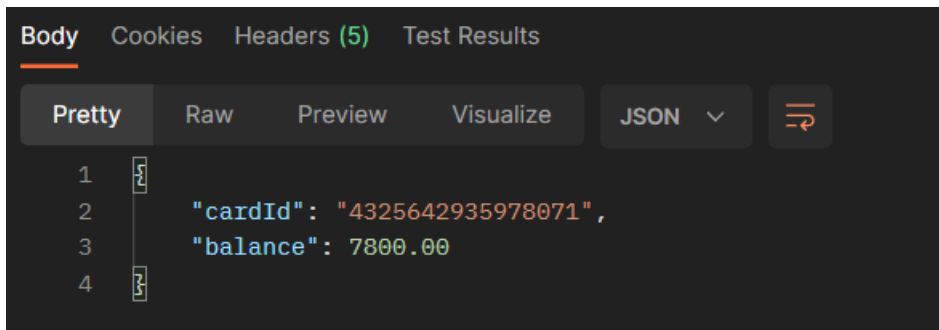
1.6) Transacción de compra



```
POST http://localhost:8080/transaction/purchase
```

¿Qué hace?: Realiza una transacción, como una compra, dependiendo del id de la tarjeta y el atributo price, que vienen en el cuerpo de la petición.

Respuesta esperada: Devuelve un objeto con el cardId y el nuevo balance o saldo.



```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "cardId": "4325642935978071",
3   "balance": 7800.00
4 }
```

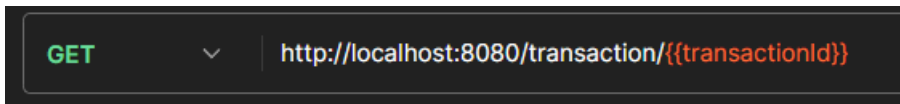
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.
- 3) Si el balance no es suficiente para realizar la compra: Retorna un mensaje de saldo insuficiente
- 4) Si la tarjeta se encuentra inactiva: Retorna un mensaje de tarjeta inactiva.

Consideraciones:

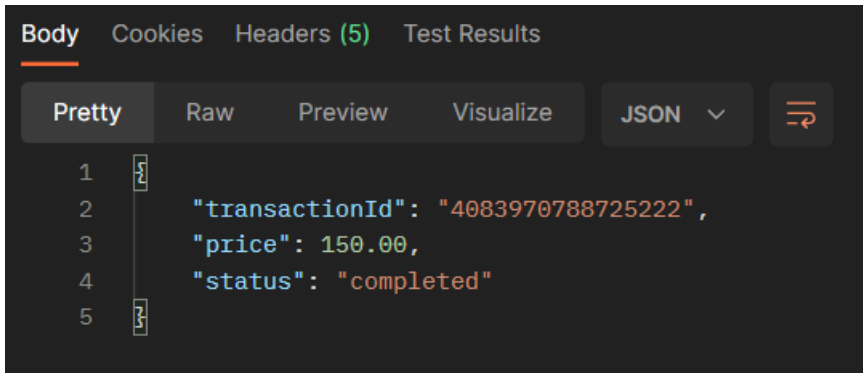
- Acá seguramente cabrían limitaciones propias de una tarjeta, por ejemplo, el número de operaciones dado cierto intervalo definido de tiempo. Limitaciones que eviten el abuso y que protejan al usuario de posibles riesgos.
- Como se mencionó anteriormente acá debería tener la contraparte también, es decir para poder establecer el dinero de donde vino y hacia donde fue.

1.7) Consultar transacción



¿Qué hace?: Consulta la transacción dependiendo del id de la transacción que vienen en el cuerpo de la petición.

Respuesta esperada: Devuelve un objeto con el Transactionid y el precio pagado, junto con el status de la transacción.



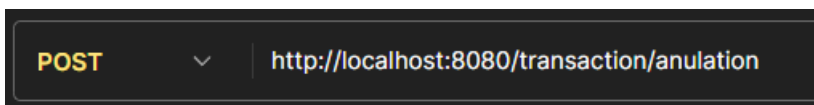
Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.
- 3) Si el transactionid no es válido: Se enviará un mensaje

Consideraciones:

Ninguna.

2.1) Anulación de transaccion

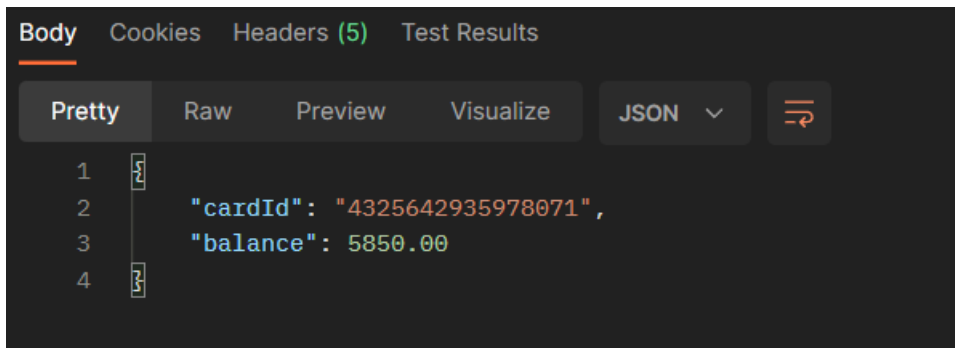


¿Qué hace?: Anula la transacción dependiendo del transactionId teniendo en cuenta TODOS los lineamientos mencionados. Es decir:

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.

Respuesta esperada: Devuelve un objeto con el cardId y el nuevo balance, es decir el retornado después de cancelar la transacción.



```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON ↕
1 {
2   "cardId": "4325642935978071",
3   "balance": 5850.00
4 }
```

Excepciones:

- 1) Si el número de la tarjeta no es válido: Retorna un mensaje de error, lo que evalúa es que siga un patrón de expresión regular, un String con 16 dígitos numéricos.
- 2) Si la tarjeta tiene 16 dígitos, pero no se encuentra en la base de datos: Retorna un mensaje de error de tarjeta inválida.
- 3) Si el transactionid no es válido: Se enviará un mensaje manifestándolo.
- 4) Si la tarjeta está inactiva: se enviará un mensaje de tarjeta inactiva.
- 5) Si la transacción ya fue anulada: se enviará un mensaje de transacción ya anulada.
- 6) Si ha transcurrido más de un día desde que la transacción se realizó: Se envía un mensaje diciendo que la transacción ya no se puede anular porque pasó un día.

Consideraciones:

Ninguna.

Requerimientos No obtenidos

Los requerimientos de testing unitario y despliegue en algún servicio de Cloud no fueron realizados.