Proyecto E-commerce

1.Introducción

En este documento se presenta el detalle de un proyecto desarrollado en java + spring boot y JPA implementando el patrón MVC, partiendo de una base de datos relacional de una tienda virtual se desarrolla los CRUD que serán utilizado para la creación de la página web. dentro de los crud desarrollos se podrá gestionar y modificar el stock de la tienda, consulta y agregar clientes, consultar y agregar nuevos productos y generar facturas

2.Objetivo

Poder tener centralizada toda la información referente a una tienda virtual de videojuegos dentro en una aplicación de forma ordenada, que permitirá mostrar los productos a vender, crear y gestionar cuentas y generar facturas.

3.Modelo de Negocio

Está basado en una tienda virtual de videojuegos la cual va dirigida a todo el público gamer en todo el mundo esto gracias a que es una web que permite llegar a cualquier persona que cuenta con un dispositivo que le permite conectarse a internet, la tienda virtual se encuentra enfocada en la venta de juegos y suscripciones las cuales están planteadas para ofrecer futuros beneficios y ofertas a los usuarios que se registren, el enfoque de la tienda virtual es ofrecer otra opción de compra junto a las existente en el mercado, tener variedad de juegos y precios competitivos.

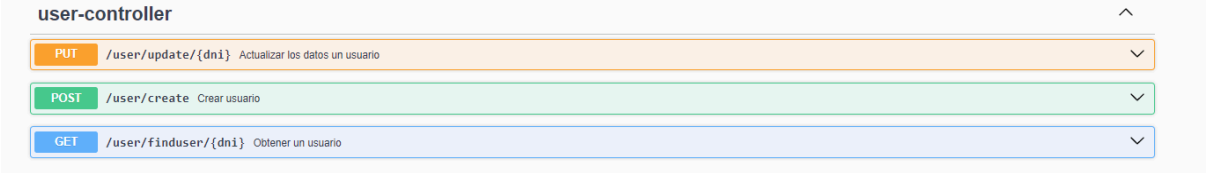
Para la documentación se utiliza la herramienta swagger que permite realizar pruebas de cada uno de los métodos creados en el desarrollo de la api.

A través del siguiente enlace se puede acceder a una interfaz que permite interactuar gráficamente con los métodos desarrollados, en cada uno de ellos se encuentra una descripción de su función.

http://localhost:8080/ecommerce/swagger-ui/index.html#/

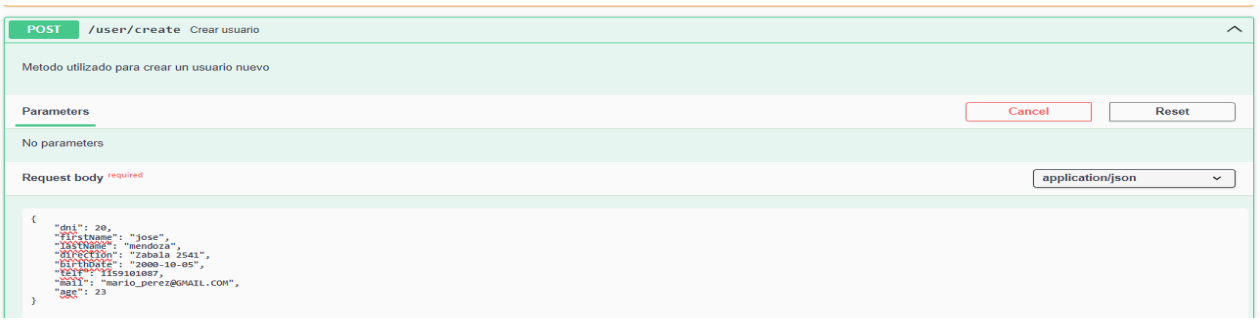
Métodos

User: Se desarrollaron 3 métodos que permiten crear, obtener y modificar los datos de un usuario o cliente, en la siguiente imagen se muestran los 3 métodos creados.

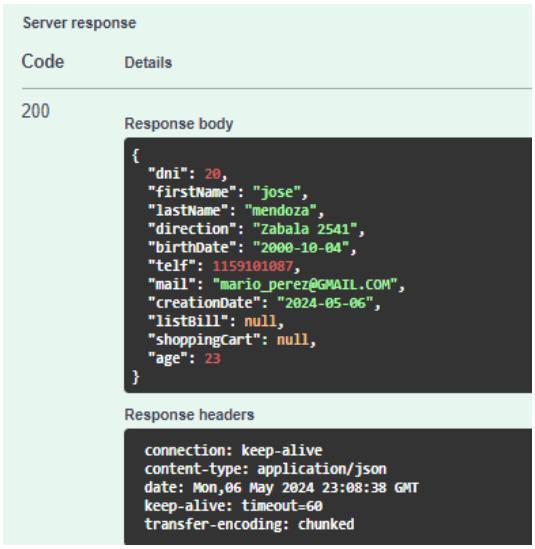


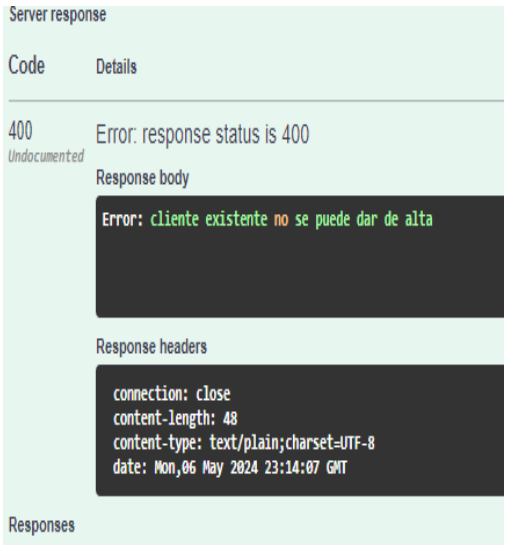
- **Post**: se utiliza para crear un cliente nuevo, en un json se envían los datos del nuevo cliente a agregar en la DB. el cual debe tener los siguiente formato.

```
{
   "dni": 20,
   "firstName": "jose",
   "lastName": "mendoza",
   "direction": "Zabala 2541",
   "birthDate": "2000-10-05",
   "telf": 1159101087,
   "mail": "mario_perez@GMAIL.COM",
   "age": 23
}
```

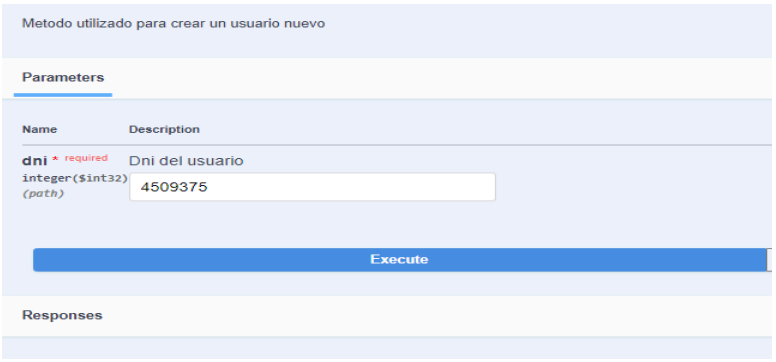


el cual retorno un estatus 200 si se registra correctamente o un estatus 400 si no se puede registrar el cliente

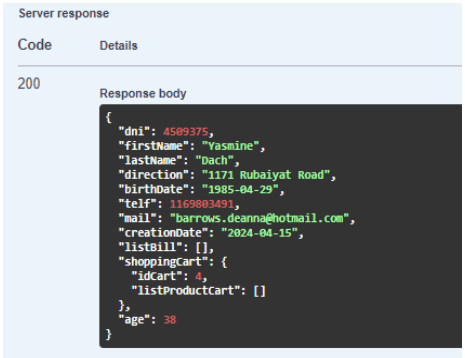




- **Get**: se utiliza para obtener los datos de un cliente, el cual se debe ingresar el id del cliente ejemplo 4509375, este ID está referenciado al DNI del cliente

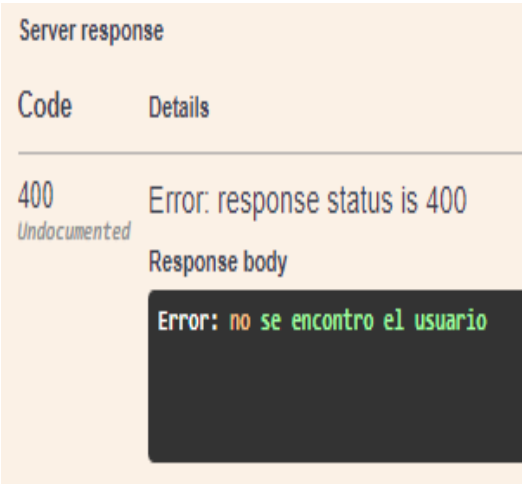


este retorno estatus 200 si obtiene los datos de cliente y 400 si no lo obtiene

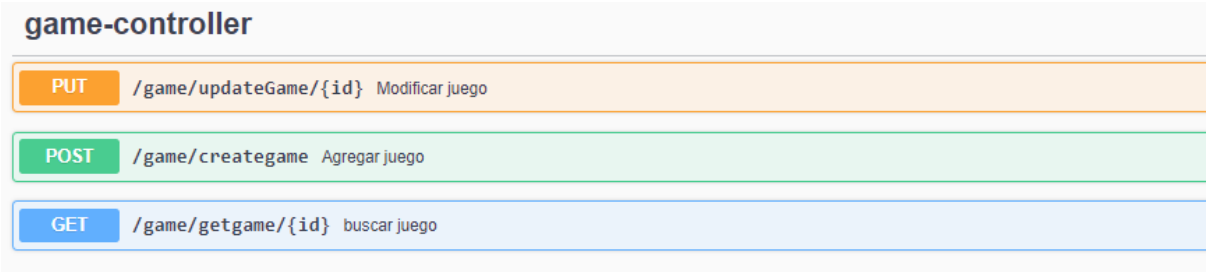


- **Put**: Se utiliza para actualizar los datos de un cliente, ingrese el DNI del mismo y los datos a actualizar

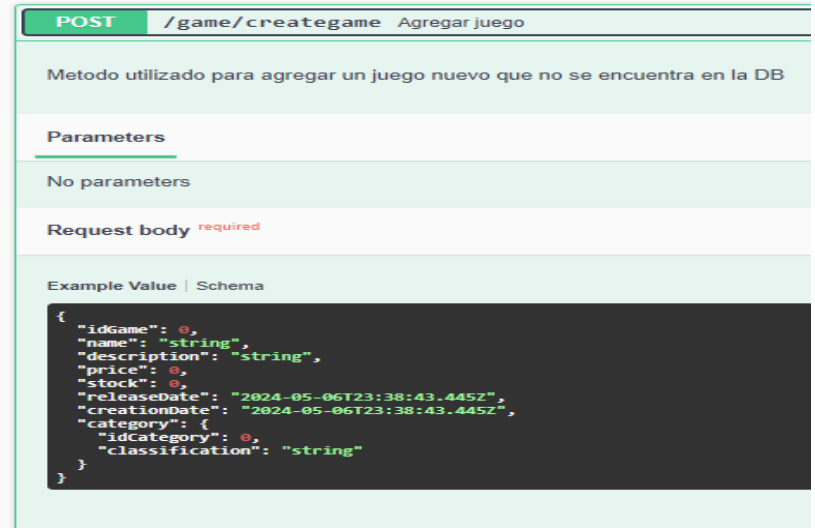
este retorna un estatus 200 si se actualiza correctamente los datos y un estatus 400 si no logra actualizar los datos



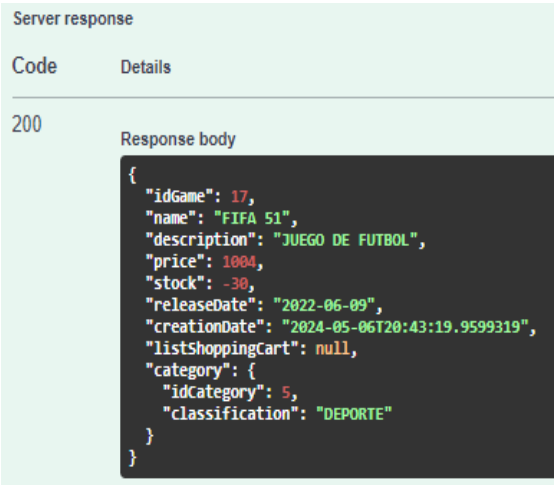
Game: Se desarrollaron 3 métodos que permiten crear, obtener y modificar los datos de un usuario o cliente, en la siguiente imagen se muestran los 3 métodos creados.

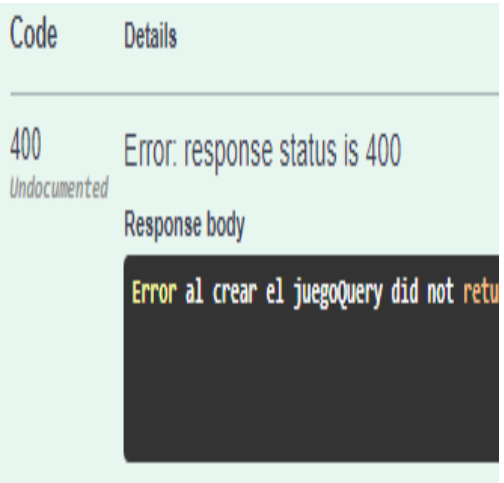


- **Post:** se utiliza para agregar un juego nuevo, en un json se envían los datos del juego a agregar en la DB. el cual debe tener los siguiente formato.

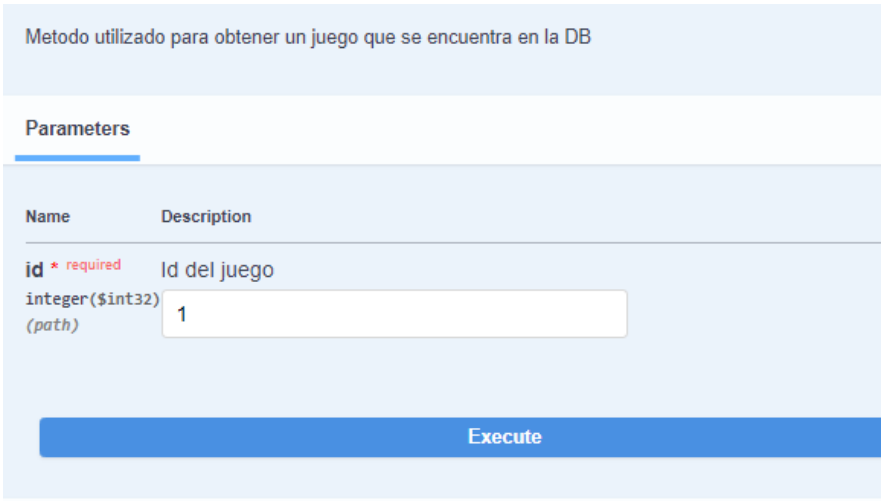


Este retorna un estatus 200 si se crea correctamente el juego y un estatus 400 si no se crea correctamente.

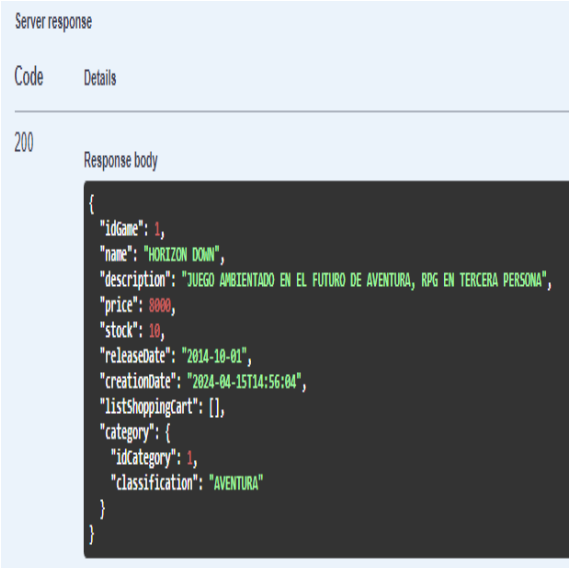


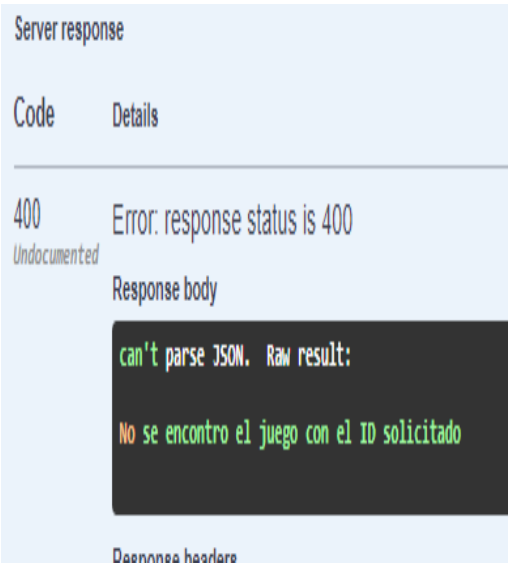


- **Get**: se utiliza para obtener los datos de un juego, el cual se debe ingresar el id del juego ejemplo 1, 2.



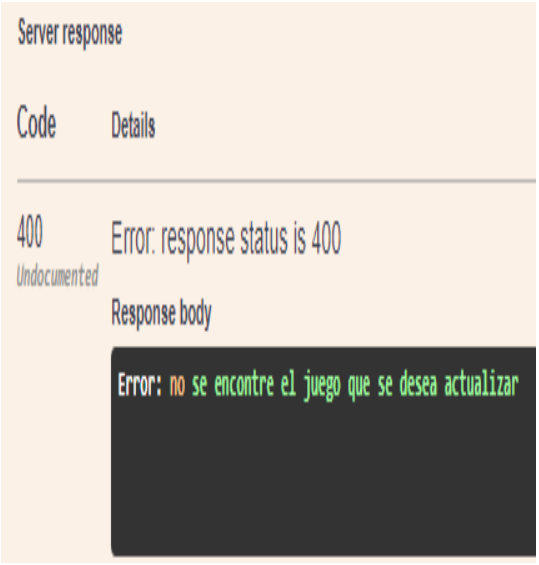
Se genera un estatus 200 si se obtiene correctamente y estatus 400 si no se encuentra el juego.



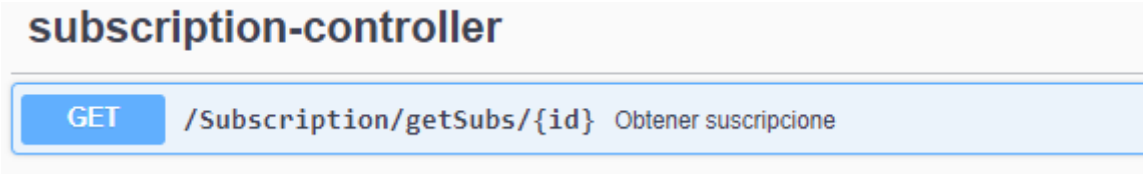


- **Put:** Se utiliza para actualizar los datos de un juego, ingresando el id del mismo y los datos a actualizar

retorna un estatus 200 si se actualiza los datos correctamente y estatus 400 si no se actualiza los datos



Subscription: Se desarrolló 1 método que permite obtener los datos de las suscripciones, en la siguiente imagen se muestran el 1 método creado.

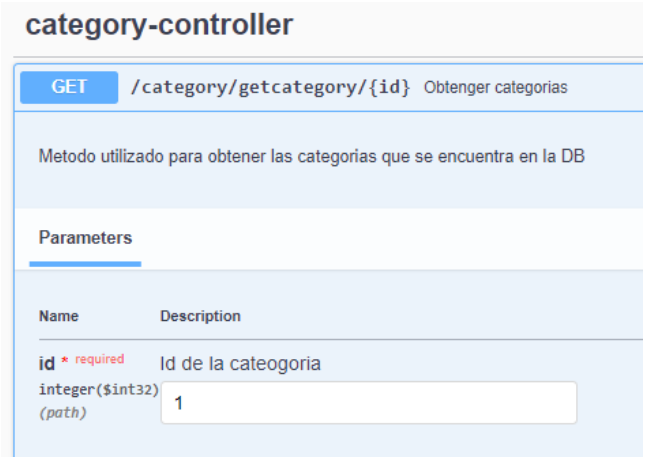


- **Get**: se utiliza para obtener los datos de la suscripción, el cual se debe ingresar el id de la suscripción ejemplo 1, 2.

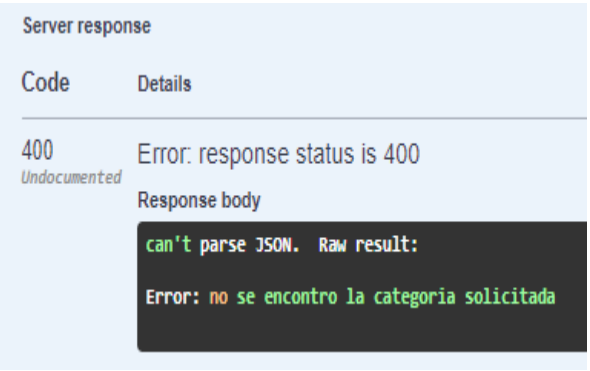
Retorna estatus 200 si se obtiene los datos correctamente y estatus 400 si no encuentra la suscripción.

Category: Se desarrolló 1 método que permite obtener los datos de las categoria, en la siguiente imagen se muestran el 1 método creado.

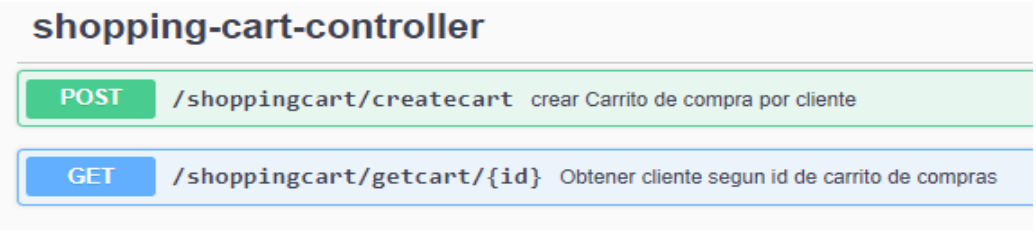
- **Get**: se utiliza para obtener los datos de la suscripción, el cual se debe ingresar el id de la categoría ejemplo 1, 2.



Retorna estatus 200 si encuentra la suscripción a buscar y estatus 400 si no encuentra la suscripción a buscar.



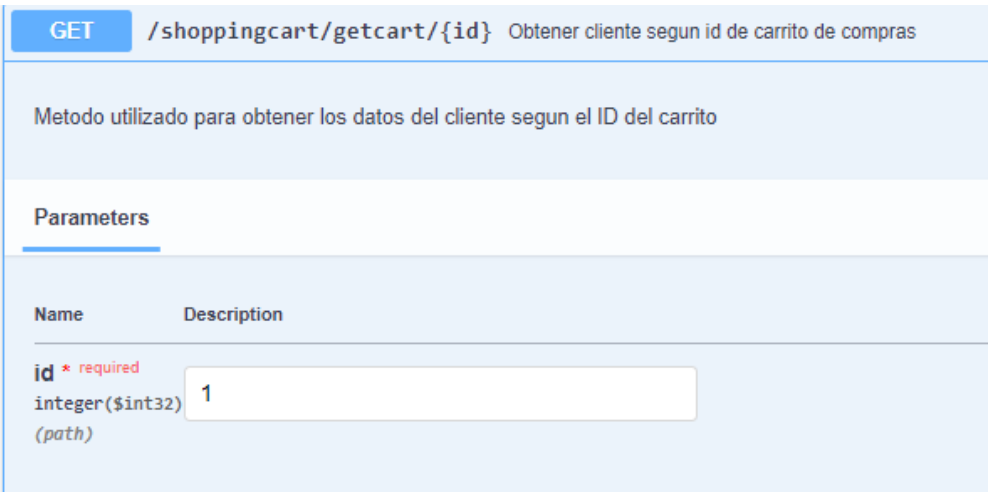
Shopping Cart: Se desarrollaron 2 métodos que permiten crear un id de carrito de compra para un cliente, obtener los datos del cliente según su id de carrito , en la siguiente imagen se muestran los 2 métodos creados.



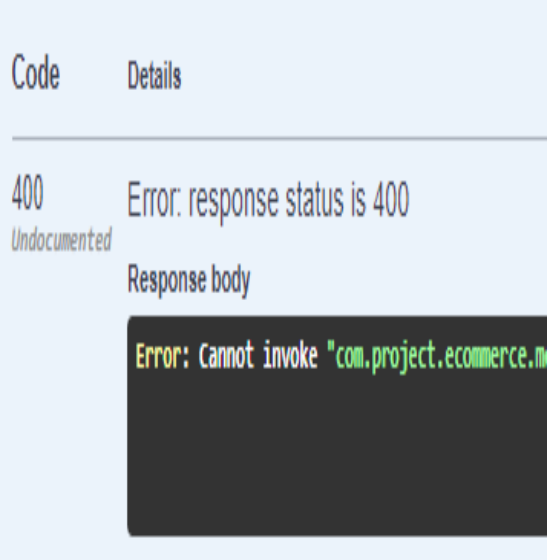
- **Post:** se utiliza para crear un carrito de comprar a un cliente, en un json se envían los datos del cliente al cual se le va generar el id del carrito de compra. el cual debe tener los siguiente formato.

Retorna estatus 200 si genera correctamente el carrito al cliente y estatus 400 si no permite generar el carrito de compra.

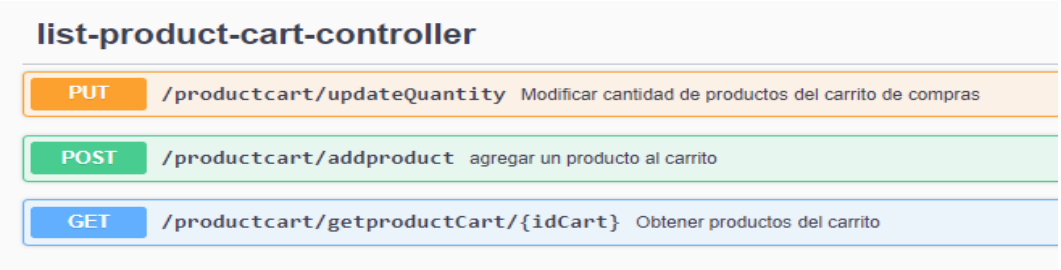
- **Get:** se utiliza para obtener los datos de cliente del id de carrito que se desee buscar



Retorna estatus 200 si consigue el carrito y a buscar y a su vez retorna un json con los datos del cliente y el id del carrito de compra que tiene asociado, en caso de no conseguir el carrito de compra solicitado retorna estatus 400 con el mensaje del error.

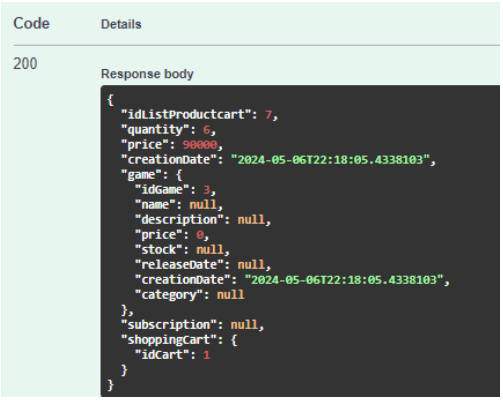


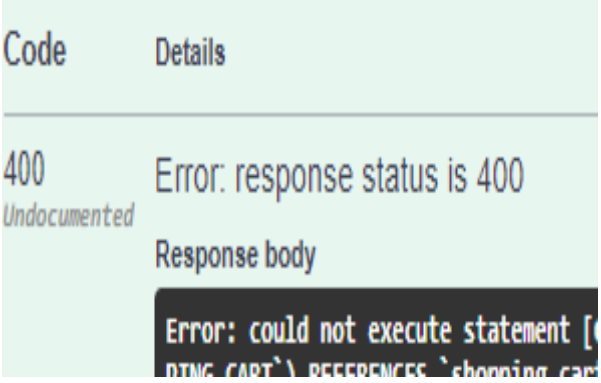
List Product Cart: Se desarrollaron 3 métodos que permite agregar un producto al carrito de compras, obtener los productos dentro del carrito de compra del cliente y modificar la cantidad de un producto dentro del carrito de compra, en la siguiente imagen se muestran los 3 métodos creados.



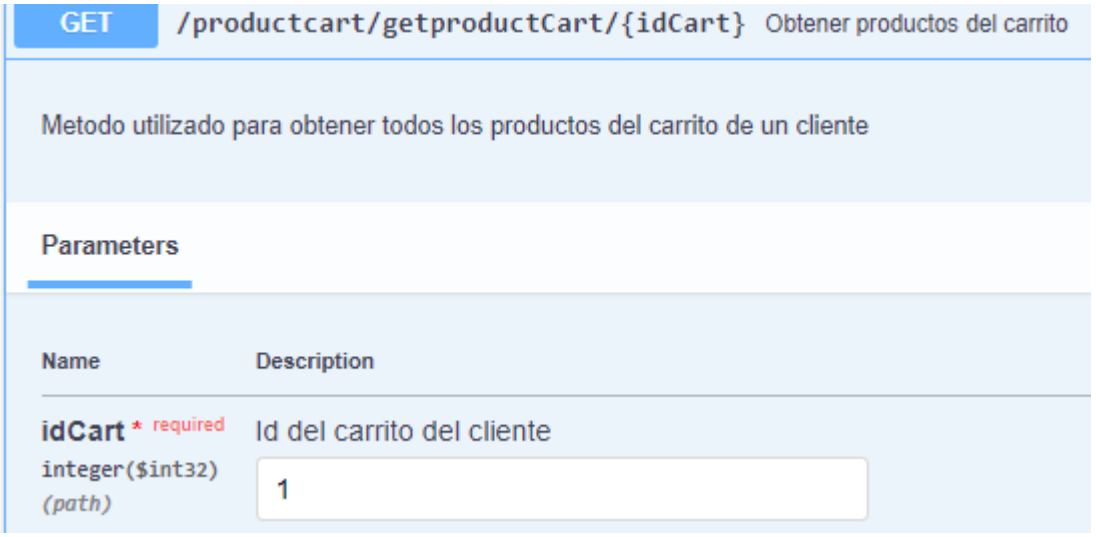
- **Post**: se utiliza para agregar un producto al carrito de compra. el cual se debe enviar a través de un json el id y tipo de producto asi como la cantidad y el id cart del cliente.

Retorna estatus 200 y un json con los datos del producto agregado y estatus 400 si no logra agregar el producto al carrito de compras.

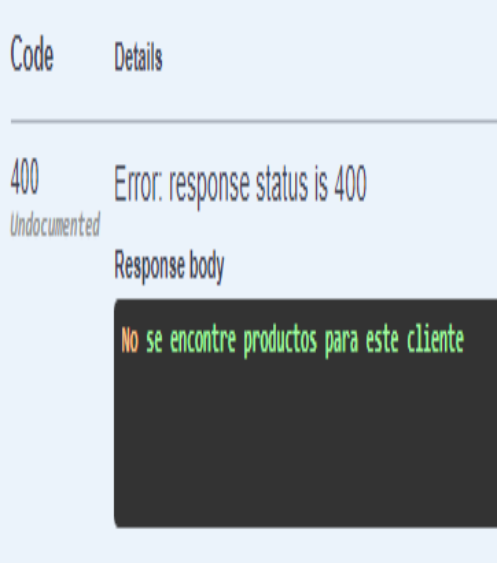




- **Get:** se utiliza para obtener los productos del carrito de compra de un cliente, se debe enviar el id del carrito de compra del cliente.



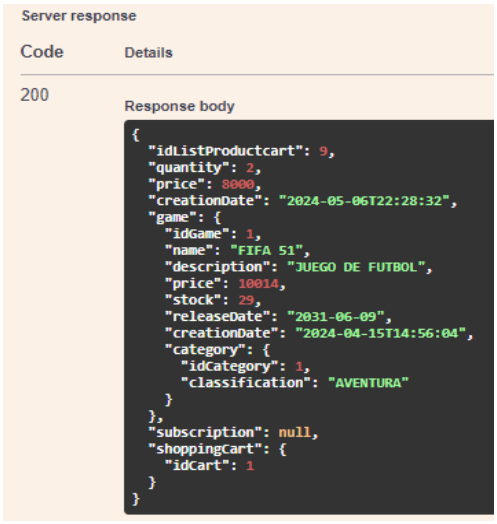
Retorna estatus 200 si consigue producto en el carrito de compra y estatus 400 si no consigue productos.

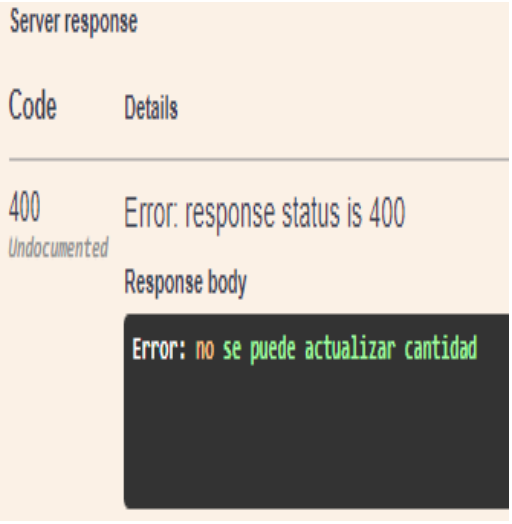


- **Put:** se utiliza para actualizar la cantidad a comprar de un producto que se encuentre en el carrito de compras del cliente

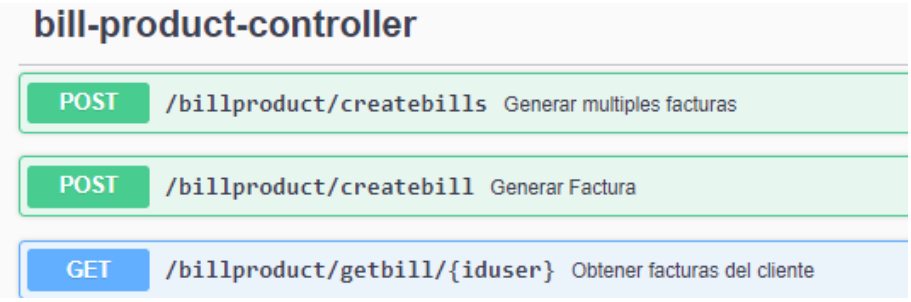
  se debe enviar en un json los datos del producto a modificar, con la cantidad y le id cart del cliente.

Retornara estatus 200 si logra modificar el producto y estatus 400 si no permite modificarlo.



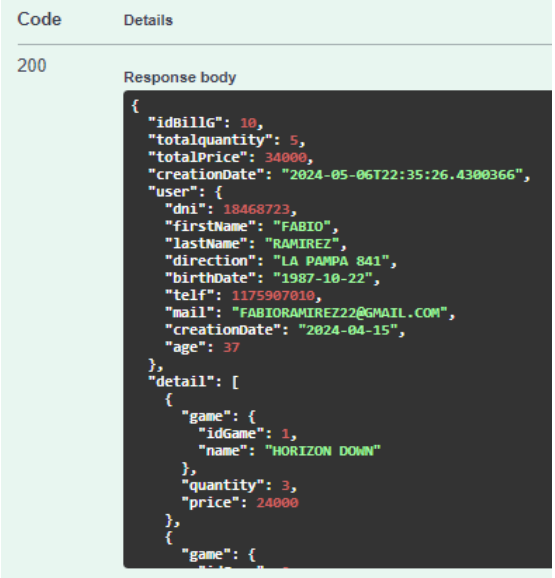


Bill Product: Se desarrollaron 3 métodos que permite generar una factura de los productos que el cliente compró, un método que permite generar multples facturas de diferentes clientes y otro para obtener todas las facturas de un cliente., en la siguiente imagen se muestran los 3 métodos creados.

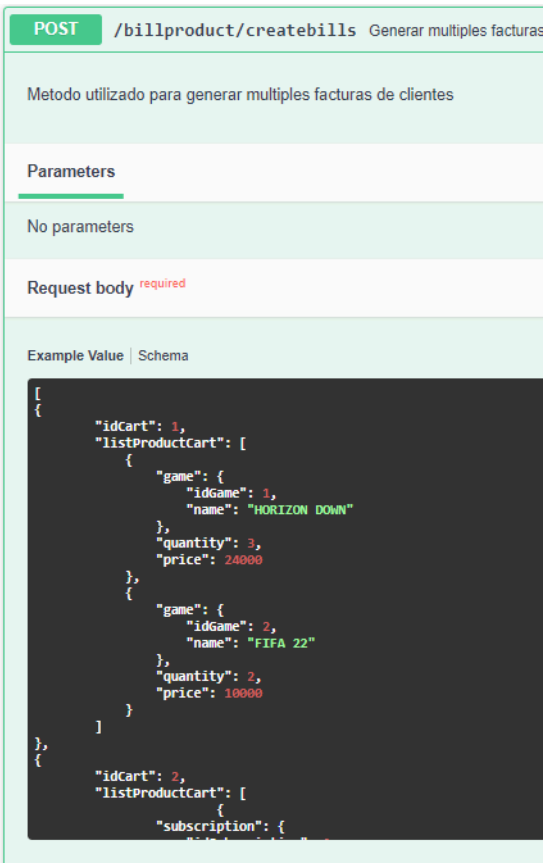


- **Post**: permite generar la factura de un cliente con los productos comprados, para esto se debe enviar en un json los datos de los productos comprados y el id del carrito de cliente

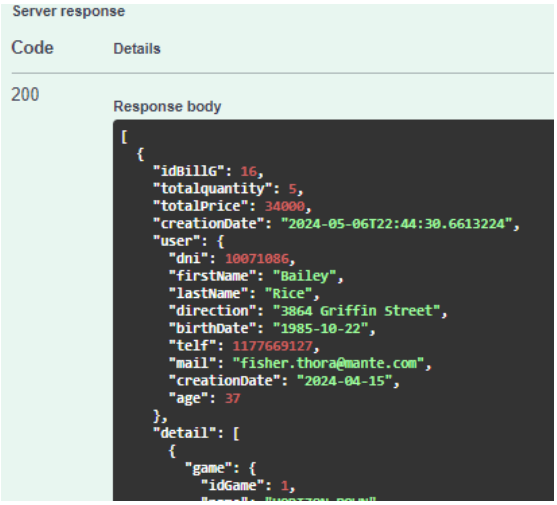
Retornara estatus 200 con un json de los datos de la factura si se genera la factura correctamente y estatus 500 si no se puede generar la factura

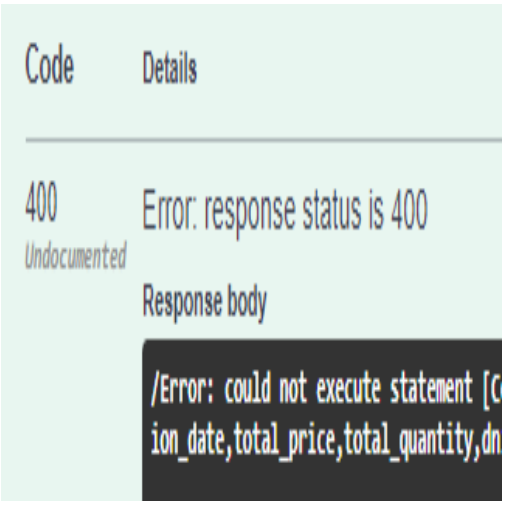


- **Post**: utilizado para generar múltiples factura de diferente clientes, de la misma forma que el método anterior se debe enviar por el json los datos de los clientes y los productos que compraron con sus precios y cantidades.

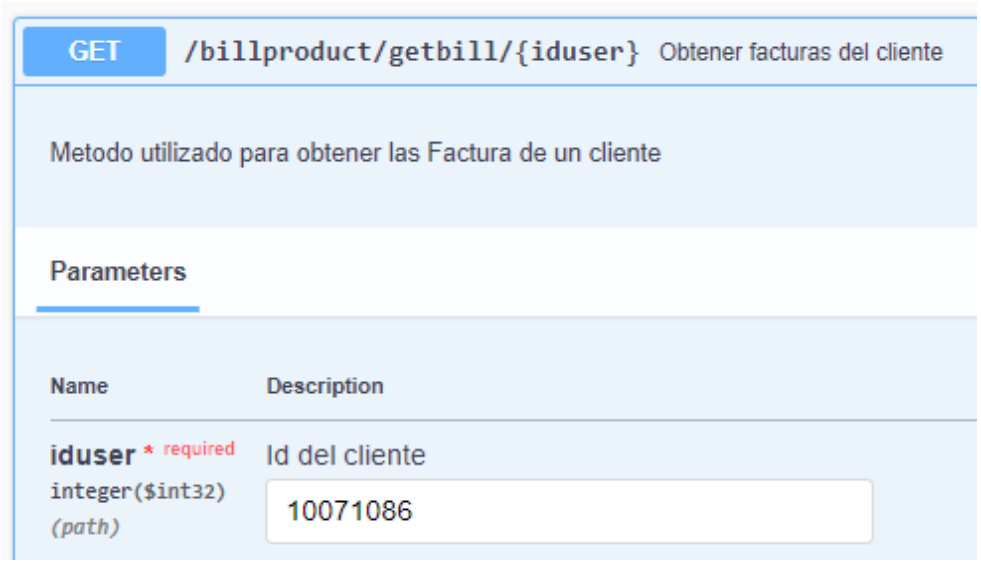


Retornara estatus 200 junto con un json con los datos de las facturas si se generan correctamente sino un estatus 400 con el mensaje del error si no se generan correctamente





- **Get**: se utiliza para obtener todas las facturas de un cliente. se debe enviar el id (DNI) del cliente al cual se desea obtener las facturas.



Retornara estatus 200 con el json de todas las facturas si se obtienen correctamente sino retorna estatus 400 si no se encuentra las facturas del cliente.

Code Details

200

Response body


```
[
  {
    "idBillG": 1,
    "totalquantity": 3,
    "totalPrice": 28000,
    "creationDate": "2024-05-06T22:46:01.1226659",
    "user": {
      "dni": 10071086,
      "firstName": "Bailey",
      "lastName": "Rice",
      "direction": "3864 Griffin Street",
      "birthDate": "1985-10-22",
      "telf": 1177669127,
      "mail": "fisher.thora@mante.com",
      "creationDate": "2024-04-15",
      "age": 37
    },
    "detail": [
      {
        "game": {
          "idGame": 1,
          "name": "FIFA 51"
        },
        "quantity": 1,
        "price": 8000
      },
      {
```


Code Details

400

Undocumented

Error: response status is 400

Response body

```
No se encuentraron facturas
```