

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.

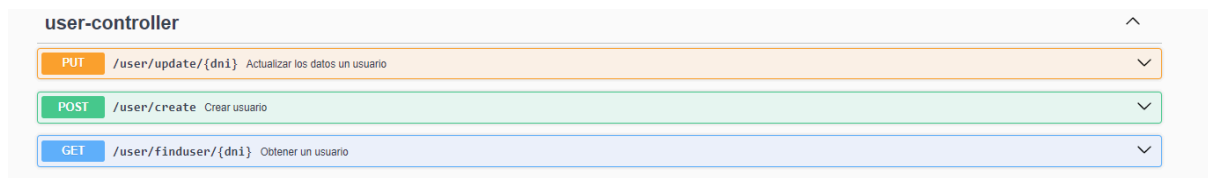
Para la documentación se utiliza la herramienta swagger que permite realizar pruebas de cada uno de los métodos creados en el desarrollo de la api.

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

Server response

Code	Details
200	<div>Response body</div> <pre>{ "dni": 20, "firstName": "jose", "lastName": "mendoza", "direction": "Zabala 2541", "birthDate": "2000-10-04", "telf": 1159101087, "mail": "mario_perez@GMAIL.COM", "creationDate": "2024-05-06", "listBill": null, "shoppingCart": null, "age": 23 }</pre> <div>Response headers</div> <pre>connection: keep-alive content-type: application/json date: Mon,06 May 2024 23:08:38 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Server response

Code	Details
400	<div>Error: response status is 400</div> <div>Undocumented</div> <div>Response body</div> <pre>Error: cliente existente no se puede dar de alta</pre> <div>Response headers</div> <pre>connection: close content-length: 48 content-type: text/plain;charset=UTF-8 date: Mon,06 May 2024 23:14:07 GMT</pre> <div>Responses</div>

- **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

Metodo utilizado para crear un usuario nuevo

Parameters

Name	Description
dni * required integer(\$int32) (path)	Dni del usuario

4509375

Execute

Responses

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

Server response

Code	Details
200	<div>Response body</div> <pre>{ "dni": 4509375, "firstName": "Yasmine", "lastName": "Dach", "direction": "1171 Rubaiyat Road", "birthDate": "1985-04-29", "telf": 1169803491, "mail": "barrows.deanna@hotmail.com", "creationDate": "2024-04-15", "listBill": [], "shoppingCart": { "idCart": 4, "listProductCart": [] }, "age": 38 }</pre>

Server response

Code	Details
400	<div>Error: response status is 400</div> <div>Undocumented</div> <div>Response body</div> <pre>ERROR: no se encontro el usuario</pre>

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

Parameters	
Name	Description
dni * required integer(\$int32) (path)	Dni del usuario 4509375

Request body
required

```
{
  "firstName": "pepito",
  "lastName": "ramirez",
  "direction": "Zabala 2541",
  "birthDate": "2000-10-05",
  "telf": "1159451087",
  "mail": "mario_perez@GMAIL.COM",
  "age": 23
}
```

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

Server response

Code	Details
200	<p>Response body</p> <pre>{ "dni": 4509375, "firstName": "pepito", "lastName": "ramirez", "direction": "Zabala 2541", "birthDate": "1985-04-29", "telf": "1169803491", "mail": "barrows.deanna@hotmail.com", "creationDate": "2024-04-15", "listBill": [], "shoppingCart": { "idCart": 4, "listProductCart": [] }, "age": 23 }</pre>

Server response

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>Error: no se encontro el usuario</pre>

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.

game-controller		
PUT	/game/updateGame/{id}	Modificar juego
POST	/game/creategame	Agregar juego
GET	/game/getgame/{id}	buscar juego

- **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.

POST /game/creategame Agregar juego

Metodo utilizado para agregar un juego nuevo que no se encuentra en la DB

Parameters

No parameters

Request body required

Example Value | Schema

```
{
  "idGame": 0,
  "name": "string",
  "description": "string",
  "price": 0,
  "stock": 0,
  "releaseDate": "2024-05-06T23:38:43.445Z",
  "creationDate": "2024-05-06T23:38:43.445Z",
  "category": {
    "idCategory": 0,
    "classification": "string"
  }
}
```

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

Server response

Code

Details

200

Response body

```
{
  "idGame": 17,
  "name": "FIFA 51",
  "description": "JUEGO DE FUTBOL",
  "price": 1004,
  "stock": -30,
  "releaseDate": "2022-06-09",
  "creationDate": "2024-05-06T20:43:19.9599319",
  "listShoppingCart": null,
  "category": {
    "idCategory": 5,
    "classification": "DEPORTE"
  }
}
```

Code

Details

400

Undocumented

Error: response status is 400

Response body

Error al crear el juegoQuery did not retu

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

Metodo utilizado para obtener un juego que se encuentra en la DB

Parameters

Name	Description
id * required integer(\$int32) (path)	Id del juego <input type="text" value="1"/>

Execute

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

Server response

Code	Details
200	<p>Response body</p> <pre>{ "idGame": 1, "name": "HORIZON DOWN", "description": "JUEGO AMBIENTADO EN EL FUTURO DE AVENTURA, RPG EN TERCERA PERSONA", "price": 8800, "stock": 10, "releaseDate": "2014-10-01", "creationDate": "2024-04-15T14:56:04", "listShoppingCart": [], "category": { "idcategory": 1, "classification": "AVENTURA" } }</pre>

Server response

Code	Details
400	<p>Error: response status is 400</p> <p>Undocumented</p> <p>Response body</p> <pre>can't parse JSON. Raw result: No se encontro el juego con el ID solicitado</pre>

Daonnnea haardare

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

Metodo utilizado para modificar los datos de un juego que se encuentra en la DB

Parameters	
Name	Description
id * required integer(\$int32) (path)	<input type="text" value="1"/>

Request body required

Example Value | Schema

```
{
  "name": "FIFA 51",
  "description": "JUEGO DE FUTBOL",
  "price": 10014.0,
  "stock": 30,
  "releaseDate": "2031-06-10",
  "idCategory": 4
}
```

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

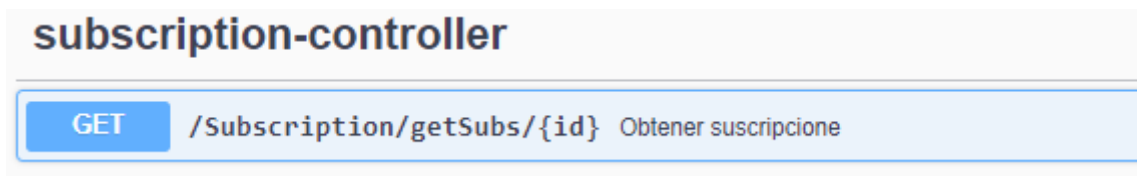
Server response

Code	Details
200	<p>Response body</p> <pre>{ "idGame": 1, "name": "FIFA 51", "description": "JUEGO DE FUTBOL", "price": 10014, "stock": 30, "releaseDate": "2031-06-09", "creationDate": "2024-04-15T14:56:04", "listShoppingCart": [], "category": { "idCategory": 1, "classification": "AVENTURA" } }</pre>

Server response

Code	Details
400	<p>Error: response status is 400</p> <p>Undocumented</p> <p>Response body</p> <pre>Error: no se encuentre el juego que se desea actualizar</pre>

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.

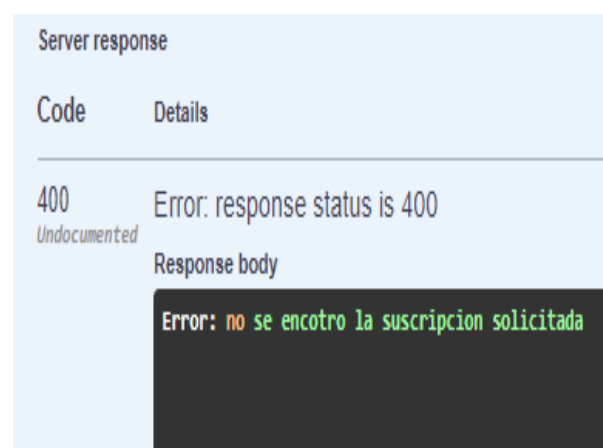
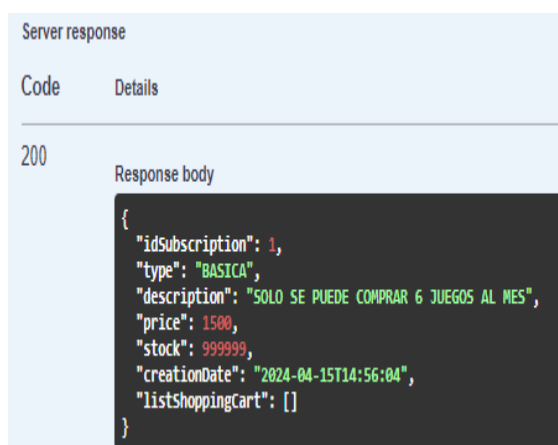
Metodo post utilizado para obtener una suscripcion

Parameters

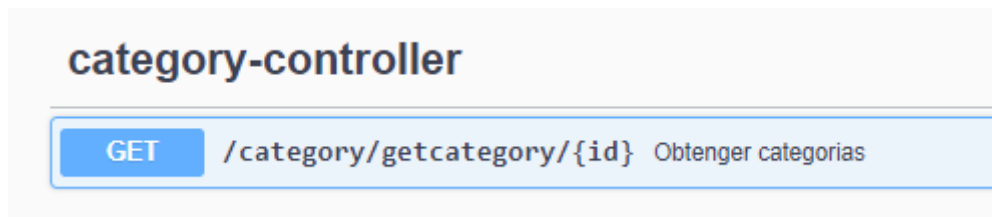
Name	Description
id * required integer(\$int32) (path)	Id de la suscripcion

1

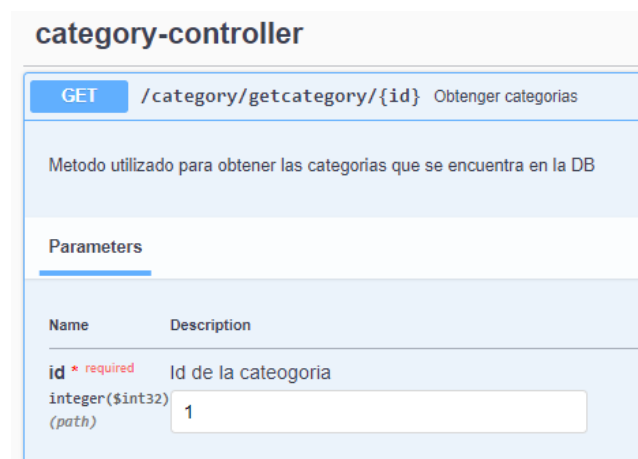
Retorna estatus 200 si se obtiene los datos correctamente y estatus 400 si no encuentra la suscripción.



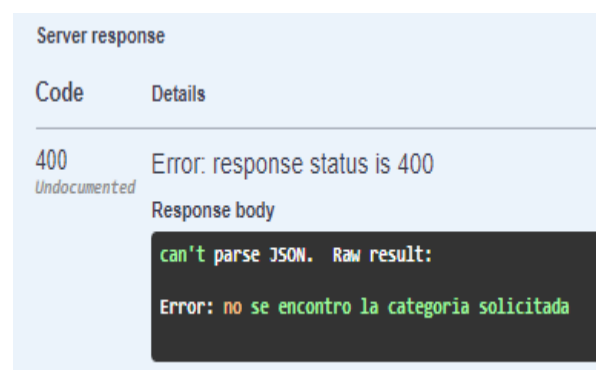
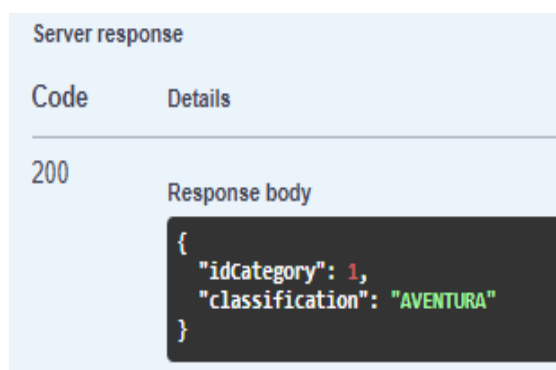
Category: Se desarrolló 1 método que permite obtener los datos de las categoria, en la siguiente imagen se muestran el 1 método creado.



- **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.

shopping-cart-controller		
POST	/shoppingcart/createcart	crear Carrito de compra por cliente
GET	/shoppingcart/getcart/{id}	Obtener cliente segun id de carrito de compras

- **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.

POST	/shoppingcart/createcart	crear Carrito de compra por cliente
Metodo utilizado para generar un ID de carrito de compra a un cliente		
Parameters		
No parameters		
Request body <small>required</small>		
Example Value Schema		
<pre>{ "user": { "dni": 4589375 } }</pre>		

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

Server response	
Code	Details
200	<div>Response body<pre>{ "idCart": 5, "user": { "dni": 2, "firstName": null, "lastName": null, "direction": null, "birthDate": null, "telf": null, "mail": null, "creationDate": "2024-05-06", "age": null } }</pre></div>

Server response	
Code	Details
400	<div>Error: response status is 400 <small>Undocumented</small> Response body<pre>El usuario ya tiene asociado un id de carrito</pre></div>

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

GET

/shoppingcart/getcart/{id} Obtener cliente segun id de carrito de compras

Metodo utilizado para obtener los datos del cliente segun el ID del carrito

Parameters

Name	Description
id * required integer(\$int32) (path)	<input type="text" value="1"/>

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.

Server response

Code	Details
200	<div>Response body</div> <pre>{ "idCart": 1, "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 } }</pre>

Code	Details
400 <i>Undocumented</i>	<div>Error: response status is 400</div> <div>Response body</div> <pre>Error: Cannot invoke "com.project.ecommerce.m"</pre>

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.

list-product-cart-controller		
PUT	/productcart/updateQuantity	Modificar cantidad de productos del carrito de compras
POST	/productcart/addproduct	agregar un producto al carrito
GET	/productcart/getproductCart/{idCart}	Obtener productos del carrito

- **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 así como la cantidad y el id cart del cliente.

POST	/productcart/addproduct	agregar un producto al carrito
Metodo utilizado para agregar un un producto al carrito de compras		
Parameters		
No parameters		
Request body <i>required</i>		
Example Value Schema		
<pre>{ "game": { "idGame": 3 }, "quantity": 6, "shoppingCart": { "idCart": 1 } }</pre>		

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.

Code	Details
200	<div>Response body</div> <pre>{ "idListProductcart": 7, "quantity": 6, "price": 50000, "creationDate": "2024-05-06T22:18:05.4338103", "game": { "idGame": 3, "name": null, "description": null, "price": 0, "stock": null, "releaseDate": null, "creationDate": "2024-05-06T22:18:05.4338103", "category": null }, "subscription": null, "shoppingCart": { "idCart": 1 } }</pre>

Code	Details
400	<div>Error: response status is 400</div> <div>Undocumented</div> <div>Response body</div> <pre>Error: could not execute statement [(SELECTING CART) REFERENCES `shopping cart`</pre>

- **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.

GET
/productcart/getproductCart/{idCart}
Obtener productos del carrito

Metodo utilizado para obtener todos los productos del carrito de un cliente

Parameters

Name	Description
idCart * required integer(\$int32) (path)	Id del carrito del cliente <input type="text" value="1"/>

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

Code	Details
200	Response body <pre>[{ "gameDto": { "idGame": 3, "name": "HORIZON FORBIDDEN WEST", "description": "juego ambientado en", "price": 15000, "stock": 32, "releaseDate": "2023-02-06", "creationDate": "2024-04-15T14:56:0", "category": { "idCategory": 1, "classification": "AVENTURA" } }, "quantityDto": 6, "priceDto": 90000, "shoppingCartDto": { "idCart": 1 } }]</pre>

Code	Details
400 <i>Undocumented</i>	Error: response status is 400 Response body <pre>No se encuentre productos para este cliente</pre>

- **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.

PUT
/productcart/updateQuantity
Modificar cantidad de productos del carrito de compra

Metodo utilizado para actualizar la cantida de un producto dentro del carrito de compras, Se debe enviar un json con los datos del producto a modificar, con la cantidad y el id del carrito.

Parameters

No parameters

Request body
required

Example Value | Schema

```

{
  "game": {
    "idGame": 1
  },
  "quantity": -1,
  "shoppingCart": {
    "idCart": 1
  }
}

```

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

Server response

Code	Details
200	<div>Response body</div> <pre> { "idListProductcart": 9, "quantity": 2, "price": 8000, "creationDate": "2024-05-06T22:28:32", "game": { "idGame": 1, "name": "FIFA 51", "description": "JUEGO DE FUTBOL", "price": 10014, "stock": 29, "releaseDate": "2031-06-09", "creationDate": "2024-04-15T14:56:04", "category": { "idCategory": 1, "classification": "AVENTURA" } }, "subscription": null, "shoppingCart": { "idCart": 1 } } </pre>

Server response

Code	Details
400	<div>Error: response status is 400</div> <div>Undocumented</div> <div>Response body</div> <pre> Error: no se puede actualizar cantidad </pre>

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

bill-product-controller		
POST	/billproduct/createbills	Generar multiples facturas
POST	/billproduct/createbill	Generar Factura
GET	/billproduct/getbill/{idUser}	Obtener facturas del cliente

- **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

POST	/billproduct/createbill	Generar Factura
Metodo utilizado para generar una factura		
Parameters		
No parameters		
Request body <small>required</small>		
Example Value Schema		
<pre>{ "idCart": 3, "listProductCart": [{ "game": { "idgame": 1, "name": "HORIZON DOWN" }, "quantity": 3, "price": 24000 }, { "game": { "idgame": 2, "name": "FIFA 22" }, "quantity": 2, "price": 10000 }] }</pre>		

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

Code

Details

200

Response body

```
{
  "idbill": 10,
  "totalquantity": 5,
  "totalPrice": 34000,
  "creationDate": "2024-05-06T22:35:26.4300366",
  "user": {
    "dni": 18468723,
    "firstName": "FABIO",
    "lastName": "RAMIREZ",
    "direction": "LA PAMPA 841",
    "birthDate": "1987-10-22",
    "telf": 1175907010,
    "mail": "FABIORAMIREZ22@GMAIL.COM",
    "creationDate": "2024-04-15",
    "age": 37
  },
  "detail": [
    {
      "game": {
        "idGame": 1,
        "name": "HORIZON DOWN"
      },
      "quantity": 3,
      "price": 24000
    },
    {
      "game": {
        "idGame": 2,
        "name": "FIFA 22"
      },
      "quantity": 2,
      "price": 10000
    }
  ]
}
```

Code

Details

500

Error: response status is 500

Undocumented

Response body

```
{
  "timestamp": "2024-05-07T01:39:20.442+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "path": "/ecommerce/billproduct/createbill"
}
```

- **Post:** utilizado para generar múltiples factura de diferentes 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.

POST

/billproduct/createbills

Generar multiples facturas

Metodo utilizado para generar multiples facturas de clientes

Parameters

No parameters

Request body

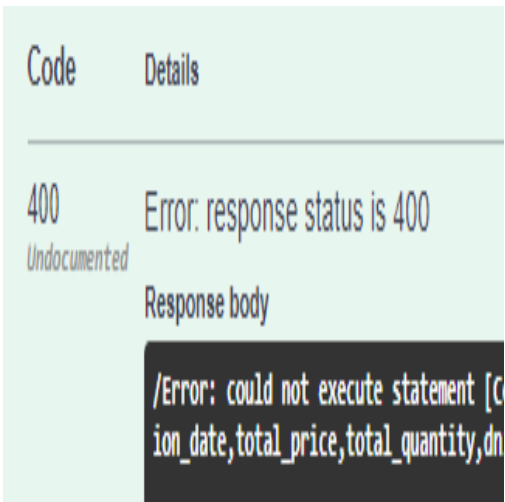
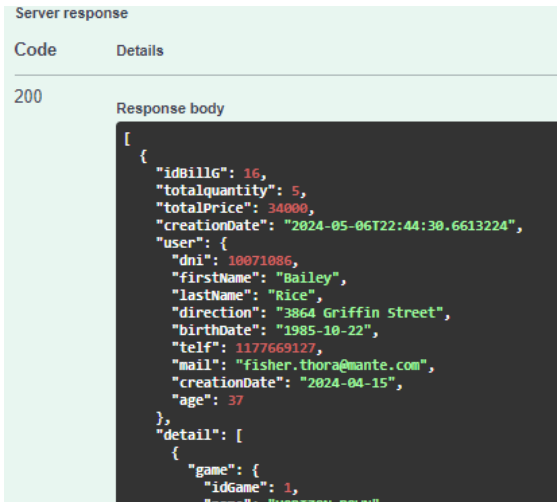
required

Example Value

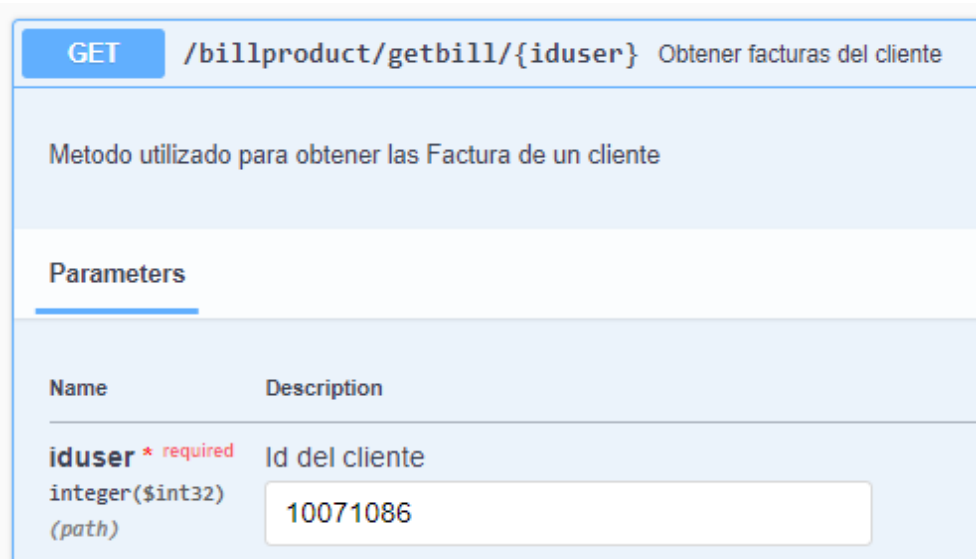
Schema

```
[
  {
    "idCart": 1,
    "listProductCart": [
      {
        "game": {
          "idGame": 1,
          "name": "HORIZON DOWN"
        },
        "quantity": 3,
        "price": 24000
      },
      {
        "game": {
          "idGame": 2,
          "name": "FIFA 22"
        },
        "quantity": 2,
        "price": 10000
      }
    ]
  },
  {
    "idCart": 2,
    "listProductCart": [
      {
        "subscription": {
          "idSubscription": 1,
          "name": "Subscription 1"
        },
        "quantity": 1,
        "price": 5000
      }
    ]
  }
]
```

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	<div>Response body</div> <pre>[{ "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 }] }]</pre>

Code	Details
400	<div>Error: response status is 400</div> <div>Undocumented</div> <div>Response body</div> <pre>No se encontraron facturas</pre>