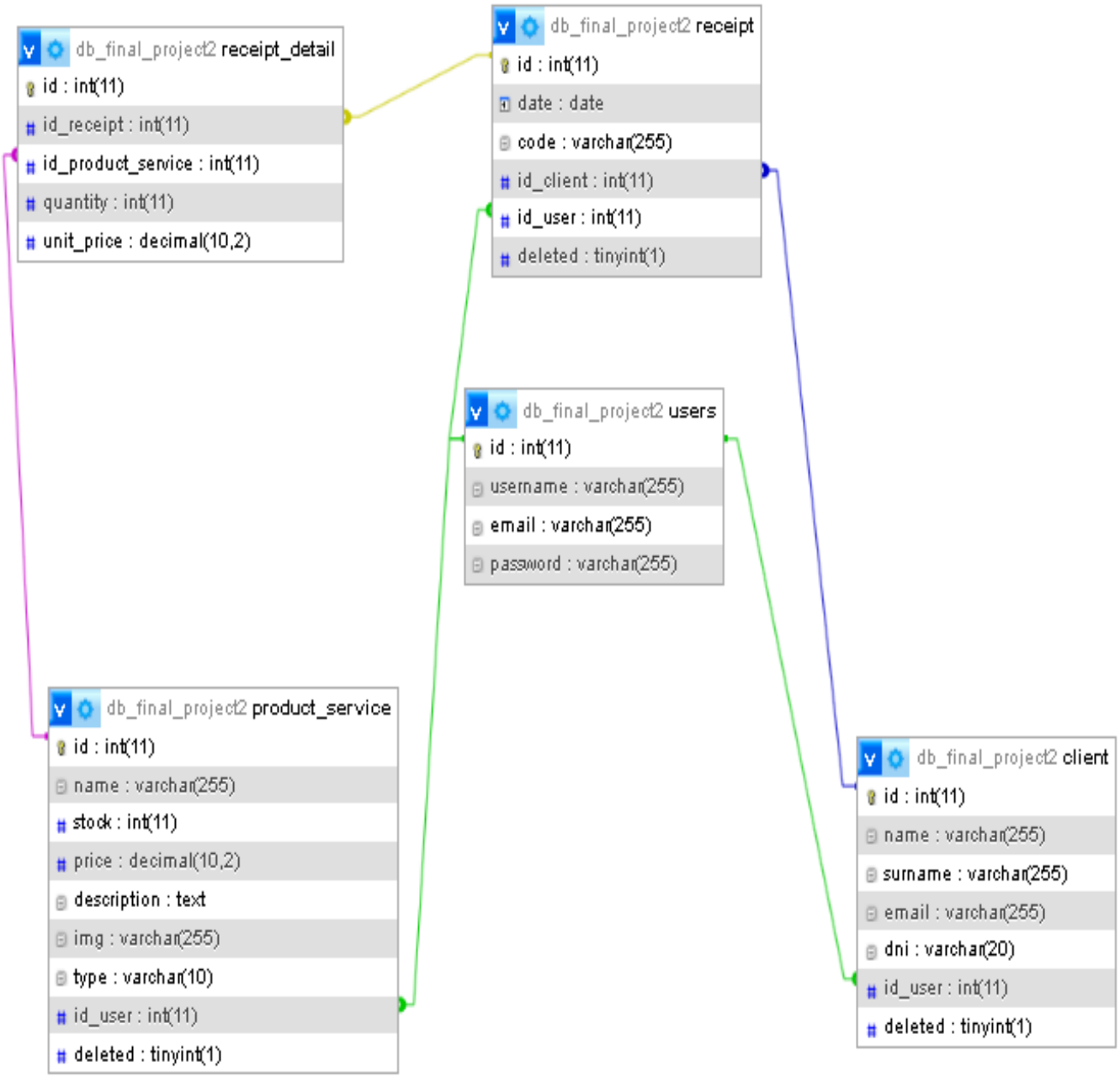
BASE DE DATOS



FRONTEND

API

LOGIN

Autentica a un usuario y emite un token JWT para las solicitudes futuras.

URL

/login

Método:

POST

Parámetros de la solicitud:

Se espera que la solicitud contenga encabezados de autorización (Authorization) con credenciales de usuario (username y password).

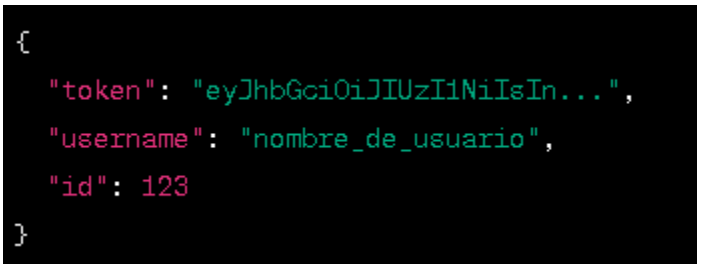
Cuerpo de la Solicitud:

No se espera un cuerpo de solicitud en formato JSON.

Respuestas:

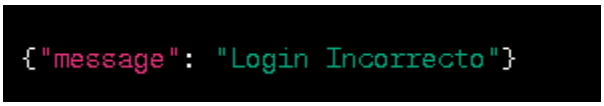
Codigo 200 OK:

Se devuelve un token JWT y la información del usuario.



Código 401 No Autorizado:

Si la autenticación falla debido a credenciales incorrectas o ausentes.



GET CLIENT BY ID

Recupera la información de un cliente específico mediante su ID.

URL

user/{id_user}/client/{id_client}

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el cliente.

{id_client}: ID del cliente que se desea recuperar.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

Se devuelve la información del cliente en formato JSON.


```
{
  "dni": "11111111",
  "email": "primerclienteusuario1@gmail.com",
  "id": 50,
  "id_user": 1,
  "name": "ClienteUsuario1",
  "surname": "Primero"
}
```


Código 404 No Encontrado:

Si no se encuentra un cliente con el ID especificado.


```
{"message": "id not found"}
```

GET ALL CLIENTS

Recupera la lista de todos los clientes pertenecientes a un usuario específico.

URL

/user/{id_user}/client

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario del que se desean obtener los clientes.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

Se devuelve una lista de objetos JSON, cada uno representando un cliente del usuario.


```
[
  {
    "dni": "11111111",
    "email": "primerclienteusuario1@gmail.com",
    "id": 50,
    "id_user": 1,
    "name": "ClienteUsuario1",
    "surname": "Primero"
  },
  {
    "dni": "22222222",
    "email": "segundoclienteusuario1@gmail.com",
    "id": 51,
    "id_user": 1,
    "name": "ClienteUsuario1",
    "surname": "Segundo"
  }
]
```

POST CLIENT

Crea un nuevo cliente y lo asocia a un usuario específico.

URL

/user/{id_user}/client

Método:

POST

Parámetros de la URL:

{id_user}: ID del usuario al que se asociará el nuevo cliente.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Datos de la Solicitud (en formato JSON):

```
{
  "name":"ClienteUsuario1",
  "surname":"Tercero",
  "email":"tercerclienteusuario1@gmail.com",
  "dni":"33333333"
}
```

Respuestas:

Código 201 Creado:

```
{
  "dni": "33333333",
  "email": "tercerclienteusuario1@gmail.com",
  "id": 52,
  "id_user": 1,
  "name": "ClienteUsuario1",
  "surname": "Tercero"
}
```

Código 400 Bad Request:

Si faltan claves requeridas en los datos de la solicitud o si los tipos de datos son incorrectos.

```
{"error": "Missing required keys"}
```

Si el tipo de datos de las claves (name, surname, email, dni) no es una cadena.

```
{"error": "Invalid data types for name, surname, email, or dni"}
```

Si el email ya está registrado para el usuario.

```
{"error": "Email already registered"}
```

Si el DNI ya está registrado para el usuario.

```
{"error": "DNI already registered"}
```

UPDATE CLIENT

Actualiza la información de un cliente existente.

URL

/user/{id_user}/client/{id_client}

Método:

PUT

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el cliente.

{id_client}: ID del cliente que se va a actualizar.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Datos de la Solicitud (en formato JSON):

```
{
  "name":"ClienteUsuario1EDITADO",
  "surname":"TerceroEDITADO",
  "email":"tercerclienteusuario1@EDITADO.com",
  "dni":"33111333"
}
```

Respuestas:

Código 201 Creado:

```
{
  "dni": "33111333",
  "email": "tercerclienteusuario1@EDITADO.com",
  "id": 52,
  "name": "ClienteUsuario1EDITADO",
  "surname": "TerceroEDITADO"
}
```

Código 400 Bad Request:

Si faltan claves requeridas en los datos de la solicitud o si los tipos de datos son incorrectos.

```
{"error": "Missing required keys"}
```

Si el tipo de datos de las claves (name, surname, email, dni) no es una cadena.

```
{"error": "Invalid data types for name, surname, email, or dni"}
```

Si el email ya está registrado para el usuario.

```
{"error": "Email already registered"}
```

Si el DNI ya está registrado para el usuario.

```
{"error": "DNI already registered"}
```

REMOVE CLIENT

Elimina lógicamente un cliente existente marcándolo como "no eliminado" en lugar de borrarlo físicamente.

URL

/user/{id_user}/client/{id_client}

Método:

DELETE

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el cliente.

{id_client}: ID del cliente que se va a eliminar.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

```
{
  "id": 52,
  "message": "deleted"
}
```

Código 404 Not Found:

Si el cliente con el ID especificado no se encuentra.

```
{
  "error": "Client not found"
}
```

GET PRODUCT / SERVICE BY ID

Obtiene los detalles de un producto o servicio específico según su ID.

URL

/user/{id_user}/product_service/{id_product_service}

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el producto/servicio.

{id_product_service}: ID del producto/servicio que se va a obtener.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

La solicitud fue exitosa, y se devuelve la información del producto/servicio.


```
{
  "description": "Pepsi Company",
  "id": 53,
  "id_user": 1,
  "img": "pepsi.jpg",
  "name": "Pepsi-co",
  "price": "1000.00",
  "stock": 100,
  "type": "Producto"
}
```


Código 404 Not Found:

Si el producto/servicio con el ID especificado no se encuentra.


```
{
  "message": "id not found"
}
```

GET ALL PRODUCT / SERVICE

Obtiene la lista de todos los productos o servicios asociados a un usuario específico.

URL

/user/{id_user}/product_service

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario para el cual se obtienen los productos/servicios.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

La solicitud fue exitosa, y se devuelve la lista de productos/servicios asociados al usuario.

```
[
  {
    "description": "Pepsi Company",
    "id": 53,
    "id_user": 1,
    "img": "pepsi.jpg",
    "name": "Pepsi-co",
    "price": "1000.00",
    "stock": 100,
    "type": "Producto"
  },
  {
    "description": "Toma lo bueno",
    "id": 54,
    "id_user": 1,
    "img": "coca.png",
    "name": "Coca Cola",
    "price": "1500.00",
    "stock": 50,
    "type": "Producto"
  }
]
```


POST PRODUCT / SERVICE

Crea un nuevo producto o servicio asociado a un usuario específico.

URL

/user/{id_user}/product_service

Método:

POST

Parámetros de la URL:

{id_user}: ID del usuario para el cual se crea el producto/servicio.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Cuerpo de la Solicitud (JSON):

```
{
  "name":"FANTOCHE",
  "stock":10,
  "price":500,
  "description":"Triple",
  "img":"FNTCH.jpg",
  "type":"Producto"
}
```


Respuestas:

Código 201 Created:

La solicitud fue exitosa, y se devuelve la información del nuevo producto o servicio creado.


```
{
  "description": "Triple",
  "id": 55,
  "id_user": 1,
  "img": "FNTCH.jpg",
  "name": "FANTOCHE",
  "price": 500,
  "stock": 10,
  "type": "Producto"
}
```


Código 400 Bad Request:

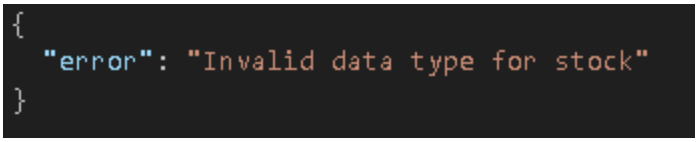
Si faltan campos requeridos.


```
{
  "error": "Missing required keys"
}
```

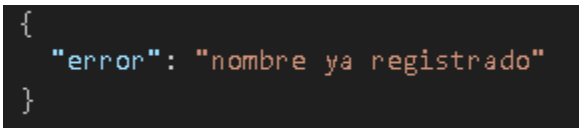

Si hay entradas de tipo invalido.


```
{
  "error": "Invalid data types for name, description, img, or type"
}
```

```
{
  "error": "Invalid data type for price"
}
```



Si el nombre ya esta registrado.



UPDATE PRODUCT / SERVICE

Actualiza la información de un producto o servicio existente.

URL

/user/{id_user}/client/{id_product_service}

Método:

PUT

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el producto o servicio.

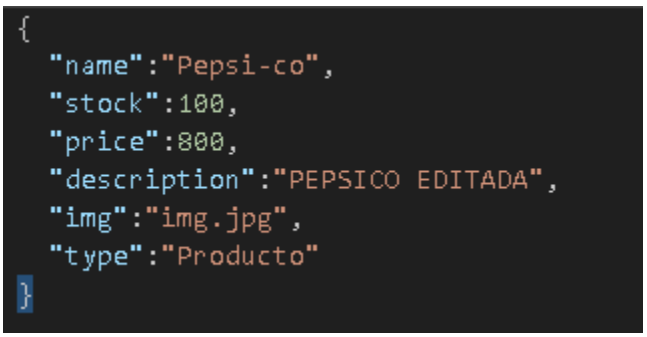
{id_product_service}: ID del producto o servicio que se va a actualizar.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Cuerpo de la Solicitud (JSON):



Respuestas:

Código 201 Created:

La solicitud fue exitosa, y se devuelve la información del nuevo producto o servicio creado.


```
{
  "description": "PEPSICO EDITADA",
  "id": 53,
  "img": "img.jpg",
  "name": "Pepsi-co",
  "stock": 100,
  "type": "Producto"
}
```


Código 400 Bad Request:

Si faltan campos requeridos.


```
{
  "error": "Missing required keys"
}
```


Si hay entradas de tipo invalido.


```
{
  "error": "Invalid data types for name, description, img, or type"
}
```

```
{
  "error": "Invalid data type for price"
}
```

```
{
  "error": "Invalid data type for stock"
}
```


Si el nombre ya esta registrado.


```
{
  "error": "nombre ya registrado"
}
```

REMOVE PRODUCT / SERVICE

Elimina lógicamente un producto o servicio existente marcándolo como "no eliminado" en lugar de borrarlo físicamente.

URL

/user/{id_user}/product_service/{id_product_service}

Método:

DELETE

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece el producto o servicio.

{id_product_service} ID del producto o servicio que se va a eliminar.

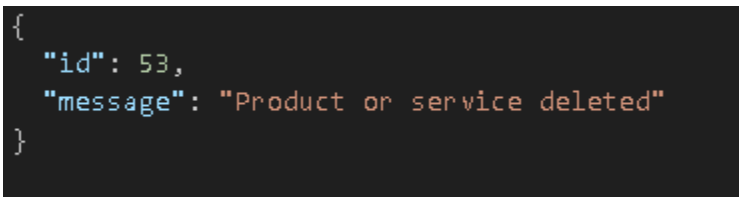
Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

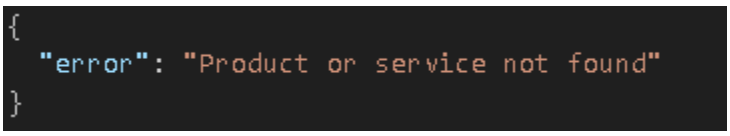
Respuestas:

Código 200 OK:



Código 404 Not Found:

Si el producto o servicio con el ID especificado no se encuentra.



GET FACTURA BY ID

Recupera la información de una factura especifica mediante su ID.

URL

user/{id_user}/receipt/{id_receipt}

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario al que pertenece la factura.

{id_receipt}: ID de la factura que se desea recuperar.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

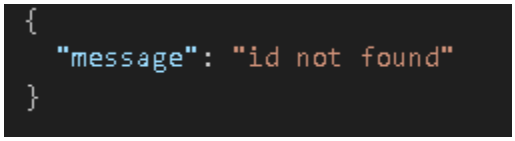
Código 200 OK:

Se devuelve la información de la factura en formato JSON.


```
{
  "code": "A",
  "date": "Sun, 31 May 2020 00:00:00 GMT
  "details": [
    {
      "id_product_service": 53,
      "name": "Pepsi-co",
      "quantity": 4,
      "type": "Producto",
      "unit_price": "800.00"
    },
    {
      "id_product_service": 55,
      "name": "FANTOCHE",
      "quantity": 3,
      "type": "Producto",
      "unit_price": "500.00"
    }
  ],
  "id": 70,
  "id_client": 50,
  "id_user": 1,
  "name_client": "ClienteUsuario1",
  "surname_client": "Primero"
}
```

Código 404 No Encontrado:

Si no se encuentra una factura con el ID especificado.



GET ALL RECEIPTS

Recupera la lista de todos las facturas pertenecientes a un usuario específico.

URL

/user/{id_user}/receipt

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario del que se desean obtener las facturas.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

Se devuelve una lista de objetos JSON, cada uno representando las facturas del usuario.

```
[
  {
    "code": "A",
    "date": "Sun, 31 May 2020 00:00:00 G
    "details": [
      {
        "id_product_service": 53,
        "name": "Pepsi-co",
        "quantity": 4,
        "type": "Producto",
        "unit_price": "800.00"
      },
      {
        "id_product_service": 55,
        "name": "FANTOCHE",
        "quantity": 3,
        "type": "Producto",
        "unit_price": "500.00"
      }
    ],
    "id": 70,
    "id_client": 50,
    "id_user": 1,
    "name_client": "ClienteUsuario1",
    "surname_client": "Primero"
  },
  {
    "code": "B",
    "date": "Sun, 31 May 2020 00:00:00 G
    "details": [
      {
        "id_product_service": 53,
        "name": "Pepsi-co",
        "quantity": 8,
        "type": "Producto",
        "unit_price": "800.00"
      }
    ],
    "id": 72,
    "id_client": 51,
    "id_user": 1,
    "name_client": "ClienteUsuario1",
    "surname_client": "Segundo"
  }
]
```


POST FACTURA

Esta ruta permite crear una nueva factura para un usuario específico.

URL

/user/{id_user}/receipt

Método:

POST

Parámetros de la URL:

{id_user}: ID del usuario para el cual se crea la factura

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Cuerpo de la Solicitud (JSON):

```
{
  "code": "B",
  "date": "2020-05-31",
  "dni_client": "22222222",
  "receipt_detail": [
    {
      "name": "Pepsi-co",
      "quantity":4
    },
    {
      "name": "Pepsi-co",
      "quantity":4
    }
  ]
}
```

Respuestas:

Código 201 Created:

La solicitud fue exitosa, y se devuelve la información de la nueva factura creada.

```
{
  "code": "B",
  "date": "2020-05-31",
  "details": [
    {
      "id_product_service": 53,
      "name": "Pepsi-co",
      "quantity": 8,
      "unit_price": "800.00"
    }
  ],
  "id": 72,
  "id_client": 51,
  "id_user": 1,
  "name_client": "ClienteUsuario1",
  "surname_client": "Segundo"
}
```


Código 400 Bad Request:

Si faltan campos requeridos.


```
{
  "error": "Missing required keys"
}
```


Si hay entradas de tipo invalido.


```
{
  "error": "Invalid data type or format for date"
}
```

```
{
  "error": "Invalid data types for code or dni_client"
}
```

```
{
  "error":"Invalid data type for receipt_detail, expected list"
}
```

```
{
  "error": "Each item in receipt_detail should be a dictionary"
}
```

```
{
  "error": "Each item in receipt_detail should have 'name' and 'quantity'"
}
```

```
{
  "error": "Invalid data types for 'name' or 'quantity' in receipt_detail"
}
```


Código 404 Not Found
Si no se encuentra el cliente del DNI en la factura.:


```
{
  "error": "Cliente no encontrado"
}
```


Si el stock de alguno de los productos es insuficiente:


```
{
  "error": "Product Pepsi-co insufficient stock"
}
```


Si un producto en la factura no se encuentra.


```
{
  "error": "Product Laptop HP not found"
}
```


GET RANKING PRODUCTOS O SERVICIOS POR CANTIDAD VENDIDA

Esta ruta permite obtener un ranking de productos y servicios basado en la cantidad vendida en todas las facturas del usuario.

URL

/user/{id_user}/receiptProductRanking

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario del que se desean obtener los productos o servicios para el ranking.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

Se devuelve un objeto con los productos y los servicios y sus cantidades vendidas.

GET RANKING CLIENTES POR CANTIDAD DE COMPRAS

Esta ruta permite obtener un ranking de clientes basado en el número de facturas realizadas por cada cliente.

URL

/user/{id_user}/receiptClientRanking

Método:

GET

Parámetros de la URL:

{id_user}: ID del usuario del que se desean obtener los clientes.

Encabezados:

Se espera que la solicitud incluya un token de acceso válido en el encabezado (x-access-token).

Se espera que la solicitud incluya el ID del usuario en el encabezado (user-id).

Respuestas:

Código 200 OK:

Se devuelve una lista de objetos JSON, cada uno representando a un cliente y su cantidad de compras o facturas a su nombre.


```
[
  {
    "dni": "11111111",
    "email": "primerclienteusuario1@gmail.com",
    "id": 50,
    "id_user": 1,
    "name": "ClienteUsuario1",
    "surname": "Primero"
  },
  {
    "dni": "22222222",
    "email": "segundoclienteusuario1@gmail.com",
    "id": 51,
    "id_user": 1,
    "name": "ClienteUsuario1",
    "surname": "Segundo"
  }
]
```


Dificultades y puntos a mejorar

Frontend:

En la construcción del frontend, identifiqué oportunidades de mejora en el manejo eficiente de recursos e información. Durante el desarrollo, noté que las consultas a la API podían volverse redundantes al acceder a diferentes secciones como clientes, facturas, productos/servicios. Para optimizar esto, podría haber implementado consultas más estratégicas al momento de iniciar sesión, trayendo toda la información necesaria y estructurándola en el frontend para un manejo más eficiente de los datos. Este enfoque podría hacer el proyecto más escalable y reducir la redundancia en las consultas a la API.

Clientes:

En la gestión de clientes, se implementó la validación de la existencia de email y DNI, pero no se controló el formato de los mismos. Ambos campos solo aceptan strings, sin restricciones de formato.

Productos y Servicios:

En los campos de stock y precio, se permite ingresar solo números. Sin embargo, en el stock, se aceptan decimales, pero solo se toma la parte entera.

La validación de nombres únicos podría mejorarse utilizando códigos de barras u otros identificadores únicos en lugar de nombres.

El campo de imagen actualmente acepta solo strings, pero idealmente debería permitir la carga de imágenes.

Al seleccionar que un producto es un servicio, el campo de stock se bloquea, pero no se reinicia si ya se ingresó un valor previamente.

Facturas:

Falta un desplegable en el campo de código para seleccionar diferentes códigos o categorías de tributación.

La lista de productos y servicios agregados a la factura debería ser interactiva para permitir su eliminación o limpieza.

Restricción interactiva en el campo de cantidad para evitar valores superiores al stock.

Búsqueda adicional por cliente además de la búsqueda por fecha.

Rankings:

Falta la opción de ocultar gráficos de productos y servicios.

Podria incluir un gráfico por montos gastados para clientes, no solo por cantidad de ventas.

Backend:

Durante el desarrollo del backend, identifiqué diversas áreas que podrían mejorarse para lograr un sistema más eficiente y escalable. A continuación, detallo algunas de las principales dificultades encontradas y áreas de mejora:

Manejo de Recursos:

Hubiera deseado adquirir y aplicar patrones de diseño que facilitaran el manejo eficiente de la información en el backend.

Por limitaciones de tiempo, algunas consultas a la base de datos se realizaron directamente en las llamadas a las rutas, y esto podría mejorarse mediante un modelado más

estructurado de los objetos.

Optimización de Consultas a la Base de Datos:

Algunas consultas a la base de datos podrían ser más eficientes. Se podria optimizar estas consultas para mejorar el rendimiento general del sistema.

Manejo de Productos y Servicios:

La gestión del stock de los servicios se manejó con un valor fijo (stock 1), pero sería valioso explorar otras estrategias, como manejar servicios y productos por separado o asignar valores nulos en el campo de stock.

Modelado de Consultas:

Un error fue el uso de objetos solo para modelar las consultas GET, mientras que en las operaciones POST y UPDATE se insertaron campos directamente en la base de datos sin utilizar el objeto. Esta inconsistencia podría mejorarse para garantizar una mayor coherencia y mantenibilidad en el código.