STORE API

DOCUMENTACIÓN E INSTRUCCIONES PARA EJECUCIÓN

Repositorio GitHub

https://github.com/Esneyder98/storeApi

Documentación Postman

https://documenter.getpostman.com/view/19168398/2s93sW9FsU

ESNEYDER SAAVEDRA CARDENAS

1. Clonar el proyecto desde github
   En la terminal en la ubicación donde se ya aguardar el proyecto ejecutar el comando
   git clone https://github.com/Esneyder98/storeApi.git

```
Usuario@DESKTOP-LQF0AAE MINGW64 ~/Documents
$ git clone https://github.com/Esneyder98/storeApi.git
Cloning into 'storeApi'...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (44/44), done.
Receiving objects:  65% (44/67)used 63 (delta 19), pack-reused 0
Receiving objects: 100% (67/67), 89.84 KiB | 948.00 KiB/s, done.
Resolving deltas: 100% (20/20), done.
```

2. Ejercicios de lógica
   2.1 Ejercicio matriz de dimensión N el cual contiene números de tipo entero. Estando en la consola en la raíz del proyecto ingresamos a la carpeta **ejercicios_logica**.
   Ejecutamos el comando **node 1.matriz.js**
   Lo cual nos mostrara en consola el resultado de los casos planteados

```
ecnica/storeApi/storeApi/ejercicios_logica (main)
$ node 1.matriz.js
1
2
2
3
3
3
true
true
true
true
false
true
3
9
15
18
70
74
```

   A su vez en el archivo mencionado **1.matriz.js** se puede validar la lógica planteada para la solución del ejercicio propuesto.
   Nota: Al sumar manualmente el resultado del caso f = [[[1, 2, 3], [2, 3, 4]], [[5, 6, 7], [5, 4, 3]], [[3,5, 6], [4, 8, 3]]] se puede validar que el resultado de la suma es 74 no 66 como se sugiere en el ejercicio.

2.2. Construya un objeto que reciba como parámetro un string y reconozca mediante expresiones regulares los respectivos criterios.

Estando en la consola en la raíz del proyecto ingresamos a la carpeta **ejercicios_logica**.
Ejecutamos el comando **node 2.objeto_operaciones.js**
El cual nos mostrara en consola el resultado de los casos planteados

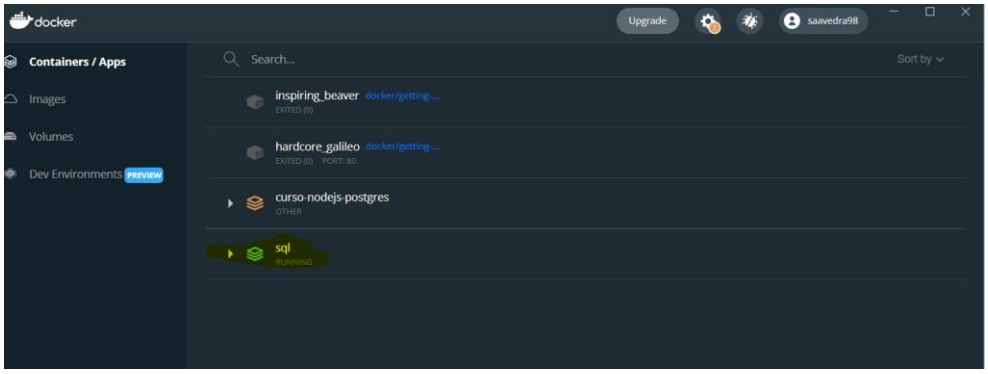
```
ecnica/storeApi/storeApi/ejercicios_logica (main)
$ node 2.objeto_operaciones.js
false
true
true
false
false
-13
-14
false
```

A su vez en el archivo mencionado **2.objeto_operaciones.js**
se puede validar la lógica planteada para la solución del ejercicio propuesto.

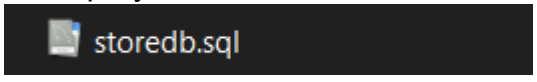
3. Configuración previa Api

- Base de datos mysql

Se debe tener descargado MySQL y algún cliente por ejemplo MySQL workbench y algún medio de conexión donde se corra la base de datos como un contenedor de Docker o xampp validamos que este corriendo correctamente.
En mi caso cree un contenedor docker



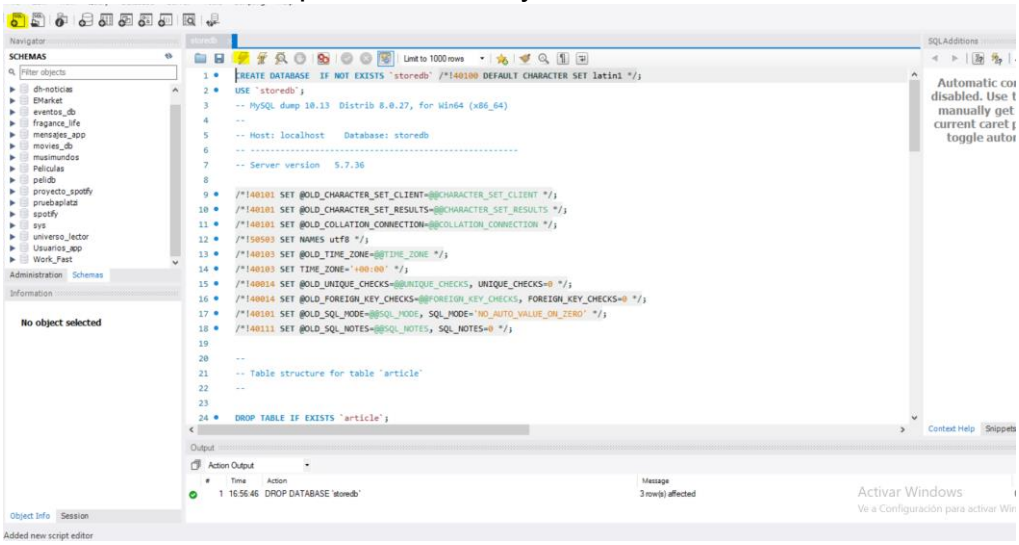
En el proyecto se encuentra la base de datos ejecutar el archivo **storedb**



En algún cliente de base de datos como workbench.

Damos en el icono 

Arrastramos el archivo storedb.sql y luego ejecutar lo cual nos creara la base de datos con sus respectivas tablas y datos iniciales



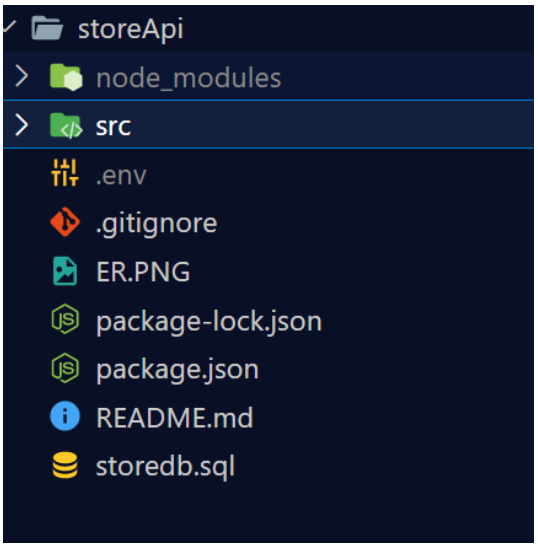
Como se observa la base de datos ha sido creada

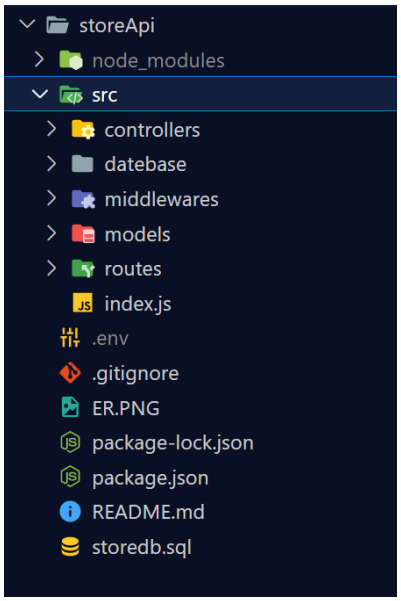
- Archivo .env (variables de entorno)

Adjunto al correo se encuentra el archivo .env el cual contiene las variables de entorno el cual por motivos de seguridad no se debe subir al repositorio de github, el cual se debe copiar y pegar en la raíz del proyecto a su vez se debe modificar según las credenciales de conexión a MySQL como lo son el usuario, la clave el nombre de la base de datos, el host , puerto y tipo de base de datos el JWT_SECRET se puede dejar el mismo o modificarlo por la clave que se desee

```
Api > .env
  USER = 'root'
  DATABASE_PASSWORD = '1234'
  DATABASE_NAME = 'storedb'
  DATABASE_HOST = '127.0.0.1'
  DATABASE_PORT = 3307
  DATABASE_DIALECT = 'mysql'
  JWT_SECRET ='wYAIszJ3GNkVljam98U2R7eg60ro1ZLQ'
```

Con lo cual el proyecto queda con la siguiente estructura





- Instalación de dependencias.
  En el editor de preferencia abrimos una terminal o abrimos una terminal por aparte; se debe tener instalado **nodeJS y npm**, estando en la raíz del proyecto ejecutamos el comando **npm install** el cual nos instala todas las dependencias del proyecto

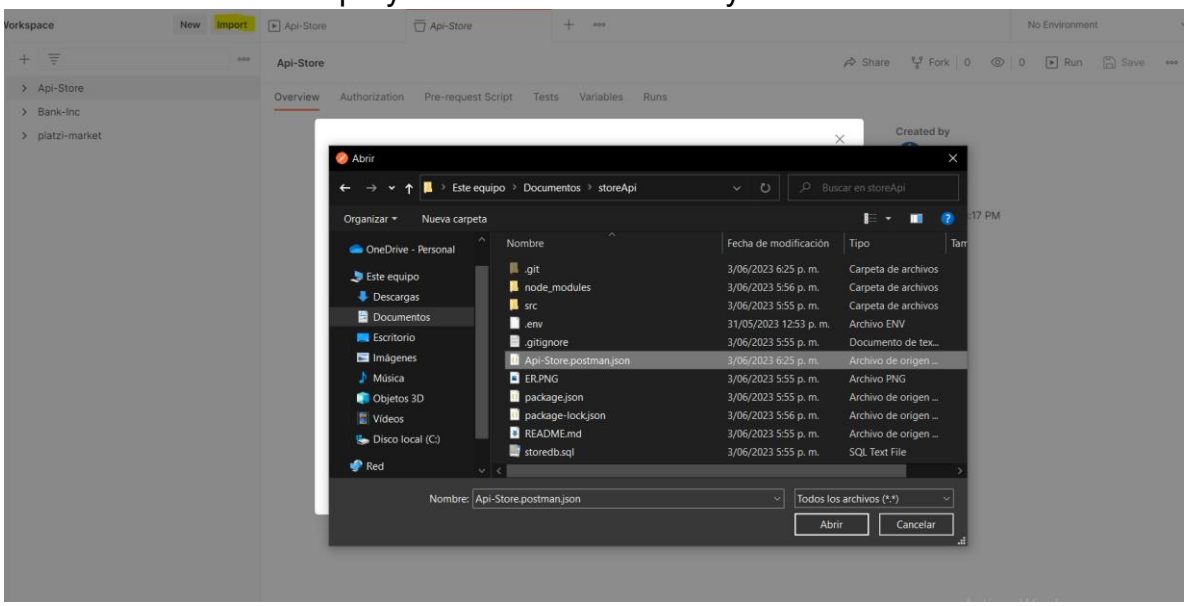
```
Usuario@DESKTOP-LQF0AAE MINGW64 ~/Documents/storeApi (main)
$ npm install

added 214 packages, and audited 215 packages in 3s

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- Ejecución del proyecto
  En la raíz del proyecto ejecutamos el comando **npm run dev**
  Si todo fue configurado correctamente ejecutara el proyecto en el puerto 3001

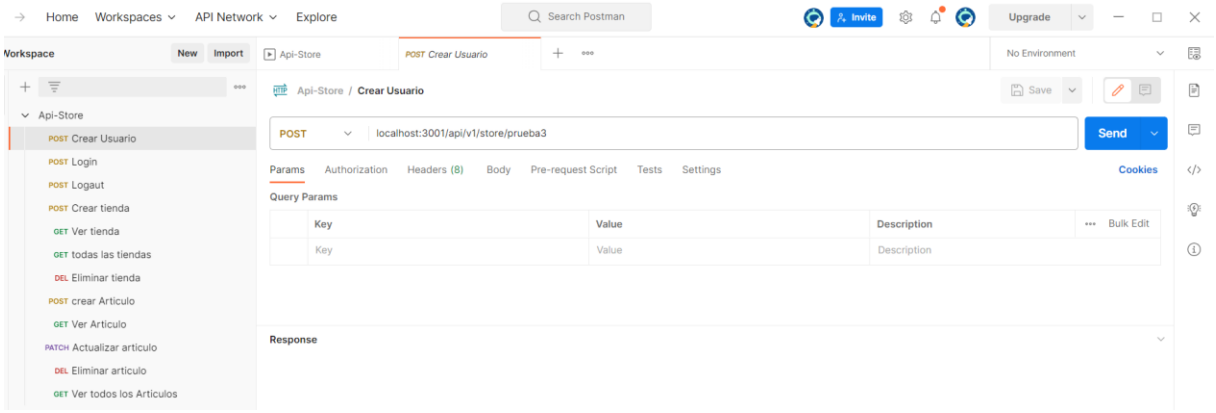
```
Usuario@DESKTOP-LQF0AAE MINGW64 ~/Documents/storeApi (main)
$ npm run dev

> storeapi@1.0.0 dev
> nodemon src/index

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
server is listening on 3001
```

- Importar colletion de enpoints en postman

Abrimos postman le damos click en importar luego files y buscamos la colección en la raíz del proyecto la seleccionamos y damos click en abir



La cual abrirá la colección en postman con todos los enpoints

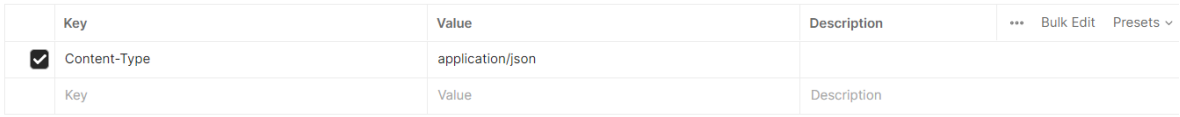


4. Documentación API

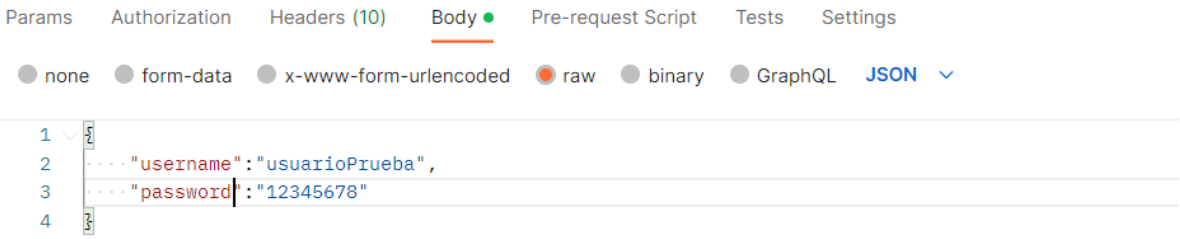
4.1. Sistema de Logueo.

- **Crear nuevo usuario**

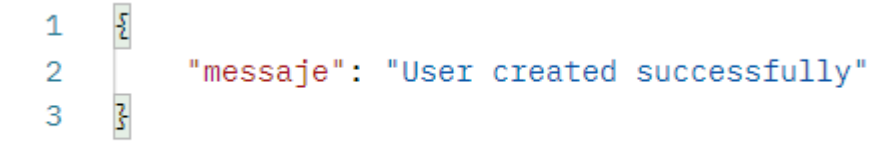
Método: POST
Enpoint: http://localhost:3001/api/v1/register
Headers



Body

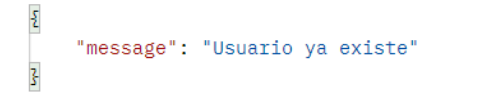


Respuesta esperada

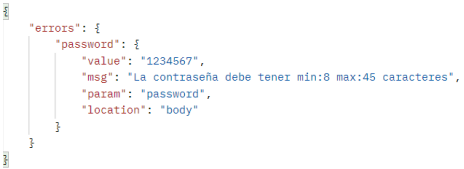


Errors y validaciones

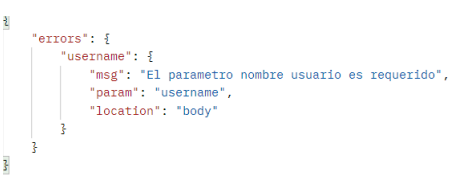
Si el usuario ya existe



Si la clave es de menos de 8 caracteres

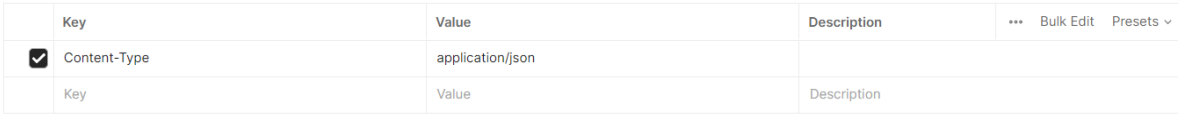


Si falta algun campo en el body

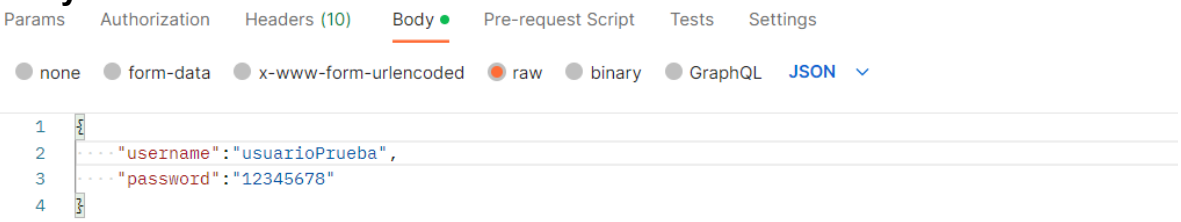


- **Loguearse con el usuario**

Método: POST
Enpoint: localhost:3001/api/v1/auth
Headers



Body



Respuesta esperada


```
{
    "acces_tokend": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOjYsInJvbGUiOiJhZG1pbiIsImlhdCI6MTY4NTgzNzc0Nn0.
        zpjc3w8HNGvBwb5TAtiXLrp7PPtiqfWjkjNewtVZPVs"
}
```


Errors y validaciones

Si el usuario no existe


```
{
    "message": "404 user Not Found"
}
```


Si la clave es incorrecta


```
{
    "message": "Credenciales invalidas"
}
```


Si falta algun campo en el body


```
{
    "errors": {
        "password": {
            "msg": "Debes completar el campo contraseña",
            "param": "password",
            "location": "body"
        }
    }
}
```

```
{
    "errors": {
        "username": {
            "msg": "El parametro nombre usuario es requerido",
            "param": "username",
            "location": "body"
        }
    }
}
```

- **Cerrar sesión**

Método: GET

Enpoint: localhost:3001/api/v1/logout
Headers
No se requiere
Body
No se requiere
Respuesta esperada


```
{
    "message": "session ended",
    "logout": true
}
```


Errors y validaciones

Si no hay sesión activa


```
{
    "message": "No hay sesion activa"
}
```

- **Crear nueva tienda**
  Store/string:name>

Método: POST
Enpoint: localhost:3001/api/v1/store/Don_Prospero
Headers
No se requiere
Body
No se requiere
Respuesta esperada


```
{
    "create": {
        "id": 12,
        "name": "Don_Prospero"
    }
}
```


Errors y validaciones

Si el usuario no esta logueado


```
{
    "errors": "Debe Loguearse para poder acceder"
}
```


Si la tienda ya existe


```
{
    "errors": "ya existe tienda con ese nombre"
}
```

- **Ver nueva tienda**
  Store/string:name>

Método: GET
Enpoint: localhost:3001/api/v1/store/Don_Prospero
Headers
No se requiere
Body
No se requiere
Respuesta esperada


```
{
    "store": {
        "id": 12,
        "name": "Don_Prospero",
        "articles": []
    }
}
```


Errors y validaciones

Si el usuario no esta logueado


```
{
    "errors": "Debe Loguearse para poder acceder"
}
```


Si la tienda no existe


```
{
    "error": "no exite tienda con ese nombre"
}
```

- **Eliminar tiendas**
  Nota: para eliminar una tienda previamente se deben eliminar los artículos que tenga asociados
  /store/<string:name

  Método: DELETE
  Enpoint: localhost:3001/api/v1/store/prueba3
  Headers
  No se requiere
  Body
  No se requiere
  Respuesta esperada

```
{
    "message": "1 tienda eliminada corectamente"
}
```

  Errors y validaciones
  Si el usuario no esta logueado

```
{
    "errors": "Debe Loguearse para poder acceder"
}
```

  Si la tienda no existe

```
{
    "error": "No exite la tienda solicitada"
}
```

  Si la tienda tiene articulos creados

```
{
    "error": "Existen articulos asociados a esta tienda "
}
```

- **Ver todas las tiendas creadas**
  /store

Método: GET
Enpoint: localhost:3001/api/v1/store/
Headers
No se requiere
Body
No se requiere
Respuesta esperada


```
{
    "listStore": [
        {
            "id": 7,
            "name": "prueba2",
            "articles": [
                {
                    "id": 1,
                    "name": "arroz",
                    "price": "2300",
                    "store_id": 7
                },
                {
                    "id": 3,
                    "name": "fideos gabazza",
                    "price": "1200",
                    "store_id": 7
                }
            ]
        },
```


Errors y validaciones

Si el usuario no esta logueado


```
    "errors": "Debe Loguearse para poder acceder"
```


Si no existen tiendas creadas

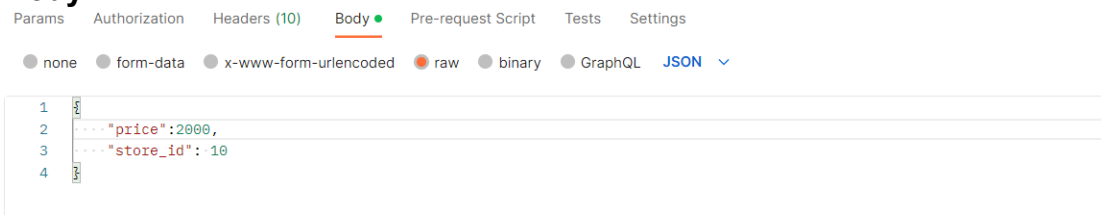

```
    "error": "no existen registros de tiendas"
```

- **Crear nuevo articulo**

article/string:name>

Método: POST
Enpoint: localhost:3001/api/v1/article/nescafe
Headers:

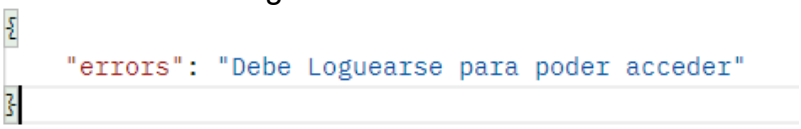
Body



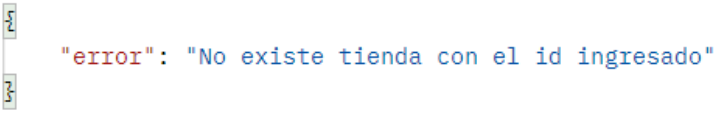
Respuesta esperada

Errors y validaciones

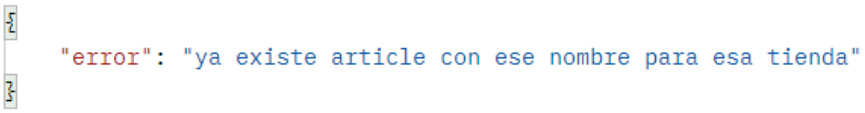
Si el usuario no esta logueado



Si la tienda no existe

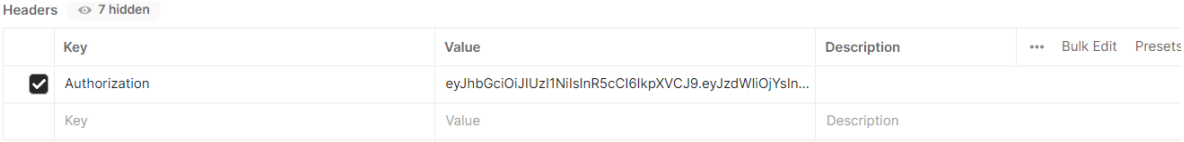


Si ya existe un articulo con ese nombre relacionado a esa tienda



- **Ver nuevo articulo**
  article/<string:name>

Método: GET
Enpoint: localhost:3001/api/v1/article/nescafe
Headers
Nota: lo copiamos de la respuesta que obtenemos al loguearnos



Body
No se requiere
Respuesta esperada


```
{
    "article": {
        "id": 5,
        "name": "nescafe",
        "price": "2000",
        "store_id": 10
    }
}
```


Errors y validaciones

Si el usuario no esta logueado


```
{
    "errors": "Debe Loguearse para poder acceder"
}
```


Si el token es incorrecto

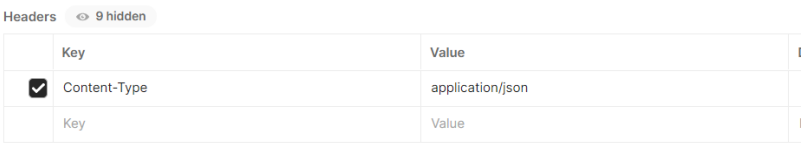

```
{
    "error": "Invalid token"
}
```


Si el articulo no existe

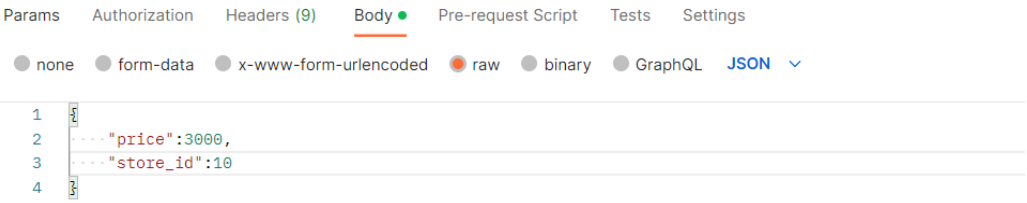

```
{
    "error": "no exite articulo con ese nombre"
}
```

- **Actualizar un articulo**
  article/<string:name>

Método: PUT
Enpoint: localhost:3001/api/v1/article/nescafe
Headers:



Body



Respuesta esperada

```
{
    "message": "Update",
    "data": {
        "id": 5,
        "name": "nescafe",
        "price": "3000",
        "store_id": 10
    }
}
```

Errors y validaciones

Si el usuario no esta logueado

```
{
    "errors": "Debe Loguearse para poder acceder"
}
```

Si el articulo no existe

```
{
    "error": "No existe articulo con el nombre: nescaf en esta tienda"
}
```

Si la tienda no existe

```
{
    "error": "Tienda ingresada no existe"
}
```

si no hay datos nuevos

```
{
    "error": "El articulo ya tiene los atributos ingresados"
}
```


Si no se envia alguno de los datos requeridos en el body


```
{
    "error": {
        "store_id": {
            "msg": "Debes completar el campo store_id",
            "param": "store_id",
            "location": "body"
        }
    }
}
```

```
{
    "error": {
        "price": {
            "msg": "Debes completar el campo precio",
            "param": "price",
            "location": "body"
        }
    }
}
```


- **Eliminar articulo**
  /article/<string:name

Método: DELETE
Enpoint: localhost:3001/api/v1/article/nescafe
Headers
No se requiere
Body
No se requiere
Respuesta esperada


```
{
    "messaje": "Article deleted"
}
```


Errors y validaciones

Si el usuario no esta logueado


```
{
    "errors": "Debe Loguearse para poder acceder"
}
```


Si el articulo no existe


```
{
    "error": "No existe articulo con ese nombre"
}
```

- **Ver todos los artículos creados**

/article

Nota: para este endpoint no se requiere que el usuario este logueado

Método: GET

Enpoint: localhost:3001/api/v1/article

Headers

No se requiere

Body

No se requiere

Respuesta esperada


```
{
    "listArticles": [
        {
            "id": 1,
            "name": "arroz",
            "price": "2300",
            "store_id": 7
        },
        {
            "id": 3,
            "name": "fideos gabazza",
            "price": "1200",
            "store_id": 7
        },
        {
            "id": 4,
            "name": "canela",
            "price": "2100",
            "store_id": 10
        }
    ]
}
```


Errors y validaciones

Si no existen articulos creados


```
{
    "error":"no existen registros de articulos"
}
```