

Documentacion

A continuación, detallare el funcionamiento de cada endpoint con imágenes y compartiré el link del documento en Postman

-Link al documento de Postman: [click al link](#)

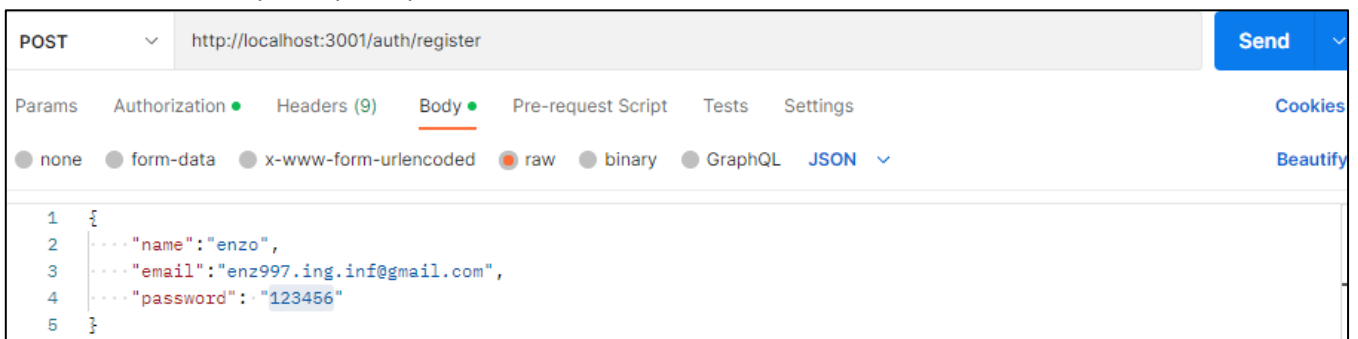
Endpoints:

-AuthRoutes (rutas de autenticación para el usuario):

1- POST - '/auth/register'.

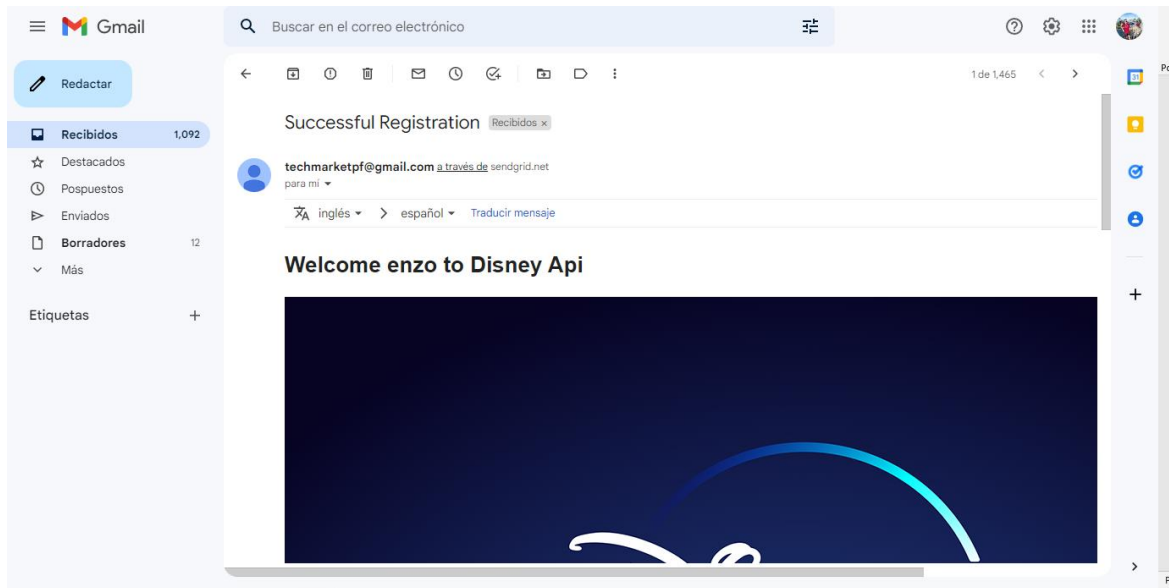
Permite crear usuarios, genera el token para poder utilizar el resto de endpoints y envia un mail de bienvenida.

Recibe por body "name", "email", "password". Valida que no haya un usuario registrado con el mismo mail y encripta la password.



Retorna en caso de éxito.



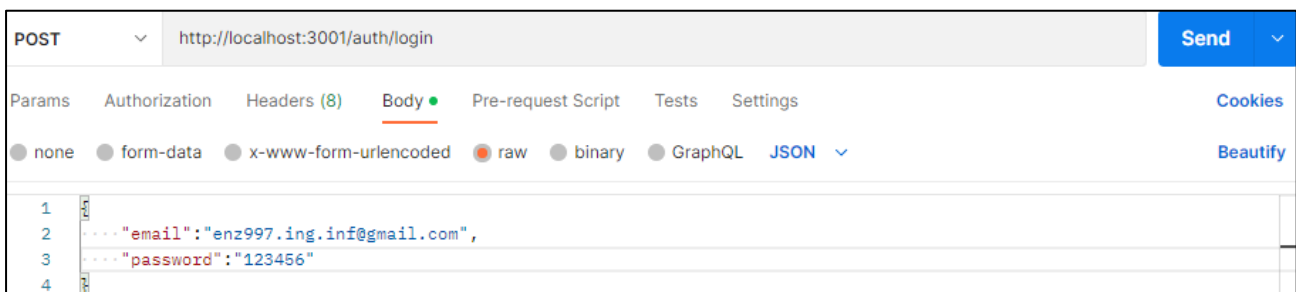


(imagen del mail de registro exitoso)

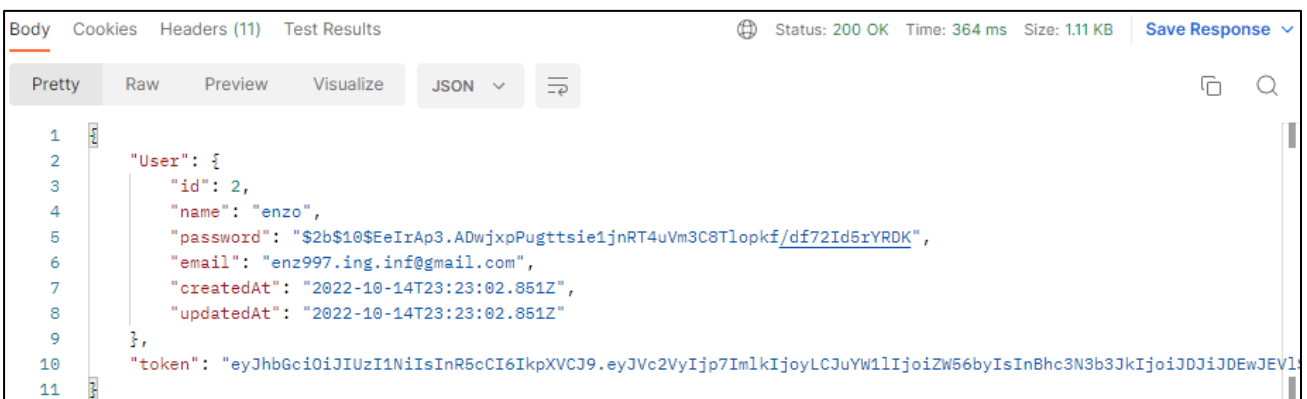
2- POST - '/auth/login'.

Permite loguear usuarios y genera token.

Recibe por body 'email' y 'password'. Valida la existencia del mail y compara la password recibida con la password encriptada en la base de datos.



Retorna en caso exitoso.



-Antes de seguir con el resto de endpoints es necesario autenticarse con el token generado previamente:

A screenshot of the Postman application window. The top bar shows the URL "http://localhost:3001/creategenre" and a "Send" button. Below the top bar are tabs for "Params", "Authorization", "Headers (9)", "Body", "Pre-request Script", "Tests", and "Settings". The "Authorization" tab is selected and highlighted with a red underline. On the left side of the "Authorization" panel, under the heading "Type", there is a dropdown menu set to "Bearer To..." and a descriptive paragraph: "The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)". On the right side of the panel, under the heading "Token", a long alphanumeric string representing a JWT token is displayed.

- GenreRoutes (crud):

Permite crear, listar, editar y borrar géneros.

1- POST - '/creategenre'.

Permite crear géneros.

Recibe por body 'name' e 'image'. Valida que el nombre no se repita.

Retorna en caso de éxito.

A screenshot of the Chrome DevTools Network tab. The 'Body' tab is selected, showing a JSON response: {"Genre": "was created successfully!"}. The status is 200 OK. The interface includes tabs for Body, Cookies, Headers (11), and Test Results. There are also buttons for Pretty, Raw, Preview, Visualize, and a JSON dropdown menu.

2- GET - '/genres'.

Permite listar géneros.

Si no hay géneros devuelve un mensaje de 'loading genres...'

Retorna en caso de éxito.

```

ty  Raw  Preview  Visualize  JSON  v  ≡
[
  {
    "id": 1,
    "name": "accion",
    "image": "hola.jpeg",
    "createdAt": "2022-10-15T00:10:37.276Z",
    "updatedAt": "2022-10-15T00:10:37.276Z"
  },
  {
    "id": 2,
    "name": "aventura",
    "image": "hola.jpeg",
    "createdAt": "2022-10-15T00:11:37.189Z",
    "updatedAt": "2022-10-15T00:11:37.189Z"
  },
  {
    "id": 3,
  }
]

```

3- GET - '/genres/:id'.

Permite buscar un género por id.

Recibe un 'id' por params. Si no hay géneros devuelve un mensaje de 'loading genres...'

Retorna en caso de éxito.

```

{
  "id": 1,
  "name": "accion",
  "image": "hola.jpeg",
  "createdAt": "2022-10-15T00:10:37.276Z",
  "updatedAt": "2022-10-15T00:10:37.276Z"
}

```

3- PUT - '/updategenre/:id'.

Permite editar un género. Recibe params el 'id' del género que se quiere editar y por body 'name' y 'image'.

```

PUT  http://localhost:3001/updategenre/8  Send
Params  Authorization  Headers (9)  Body  Pre-request Script  Tests  Settings  Cookies
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON  Beautify
1  {
2    "name": "fantasia",
3    "image": "hola.jpeg"
4  }

```

Retorna en caso de éxito.

```

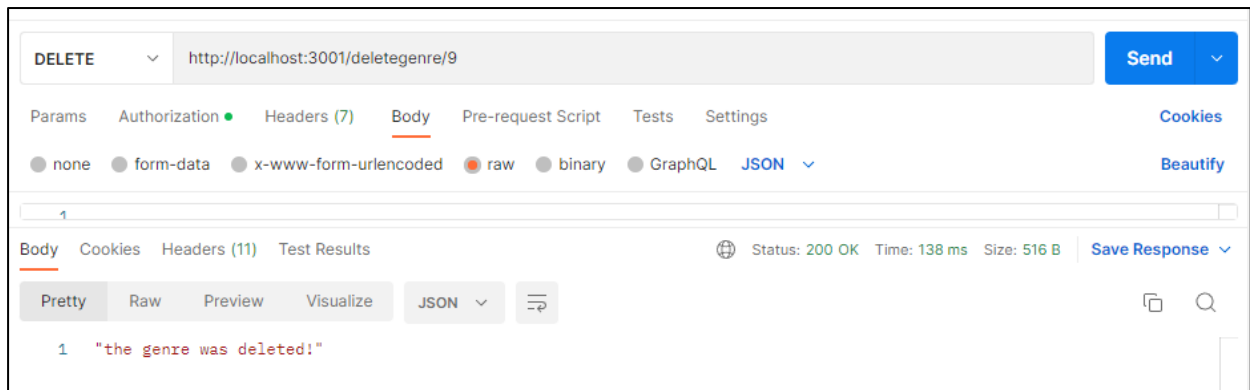
Body  Cookies  Headers (11)  Test Results  Status: 200 OK  Time: 125 ms  Size: 525 B  Save Response
Pretty  Raw  Preview  Visualize  JSON  ≡
1  "Genre was updated successfully!"

```

4- DELETE - '/deletegenre/:id'.

Permite eliminar un género.

Recibe un 'id' por params.



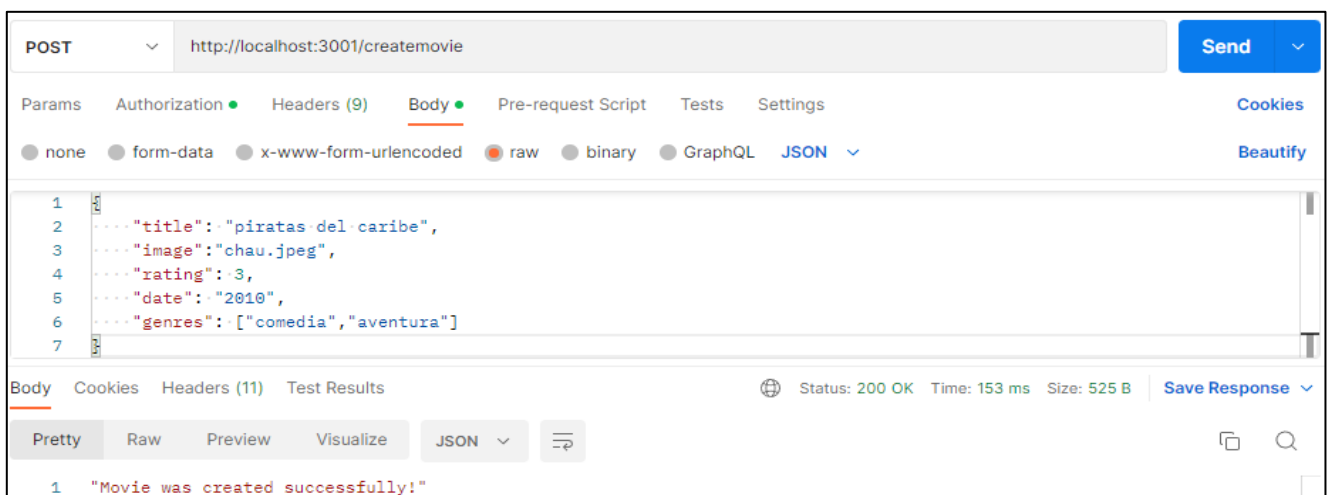
-MoviesRoutes (crud):

Permite crear, listar, buscar, filtrar, editar y borrar películas-series.

1- POST - '/createmovie'.

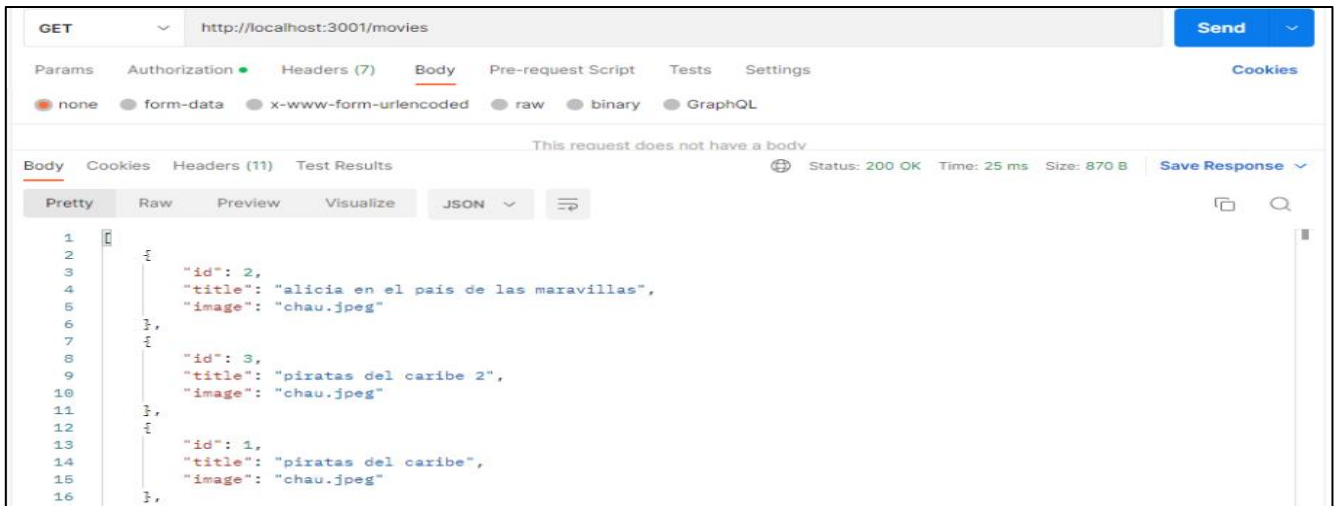
Permite crear Movies-seires.

Recibe por body 'title', 'image', 'rating', 'date' y 'genres'. Comprueba que no se repita el titulo ingresado y que exista el género.



2- GET - '/movies'.

Devuelve el listado de películas solo con los atributos 'id', 'title', 'image'.



3- GET - '/movies/id'.

Devuelve el detalle de la película cuando se le envía un 'id' valido por params. Además, se muestran los modelos relacionados como Genres y Characters.

GET http://localhost:3001/movies/2 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (11) Test Results Status: 200 OK Time: 29 ms Size: 815 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": 2,
3    "image": "chau.jpeg",
4    "title": "alicia en el país de las maravillas",
5    "date": null,
6    "rating": 3,
7    "genres": [
8      {
9        "id": 2,
10       "name": "aventura",
11       "image": "hola.jpeg"
12     },
13     {
14       "id": 4,
15       "name": "comedia",
16       "image": "hola.jpeg"
17     },
18     {
19       "id": 8,
20       "name": "fantasia",
21       "image": "hola.jpeg"
22     }
23   ],
24   "characters": [

```

4- GET - '/movies?query=query'.

Busca por nombre, filtra por genero y ordena de manera ascendente o descendente según el date (año de creación). Los parametros son recibidos por query y son combinables, en caso de no recibir parámetros trae la lista de movies.

GET http://localhost:3001/movies?name=pi&genre=com&order=ASC Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	pi			
<input checked="" type="checkbox"/>	genre	com			
<input checked="" type="checkbox"/>	order	ASC			

Body Cookies Headers (11) Test Results Status: 200 OK Time: 183 ms Size: 935 B Save Response

Pretty Raw Preview Visualize JSON

```

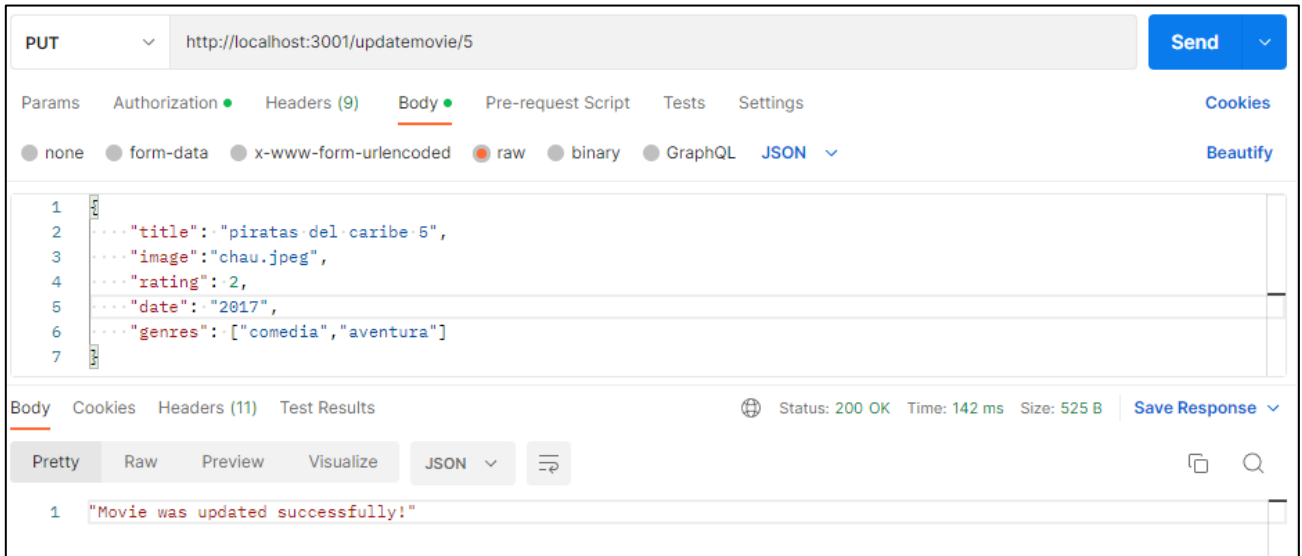
1  [
2    {
3      "id": 1,
4      "title": "piratas del caribe",
5      "image": "chau.jpeg",
6      "date": "2003",
7      "genres": [
8        {
9          "id": 4,
10         "name": "comedia"
11       }
12     ]
13   },
14   {
15     "id": 4,
16     "title": "piratas del caribe 4".

```

5- PUT - '/updatemovie'.

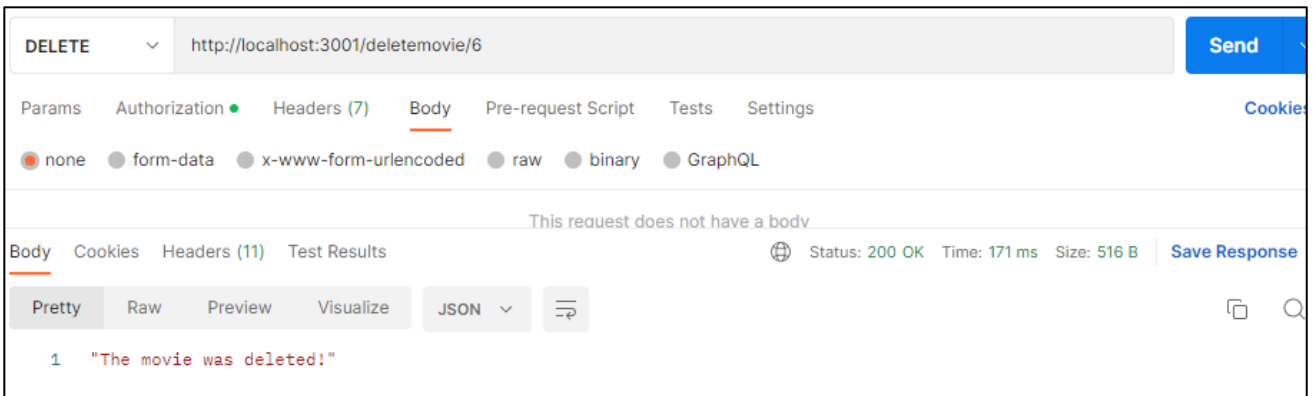
Permite editar Movies-seires.

Recibe por body 'title', 'image', 'rating', 'date' y 'genres' e 'id' por params.

**6- DELETE - '/deletemovie'.**

Permite borrar Movies-seires.

Recibe por params el 'id' y borra.

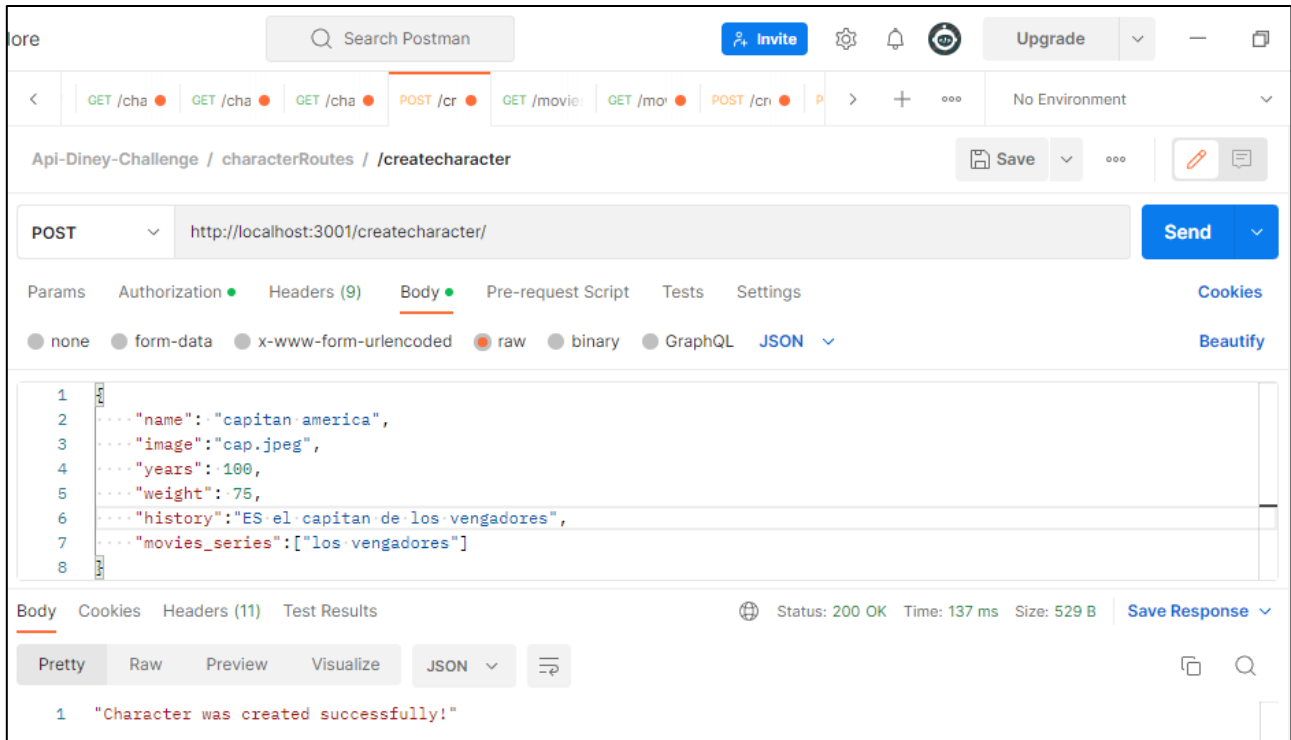
**-CharacterRoutes (crud):**

Permite crear, listar, buscar, filtrar, editar y borrar personajes.

1- POST - '/createcharacter'.

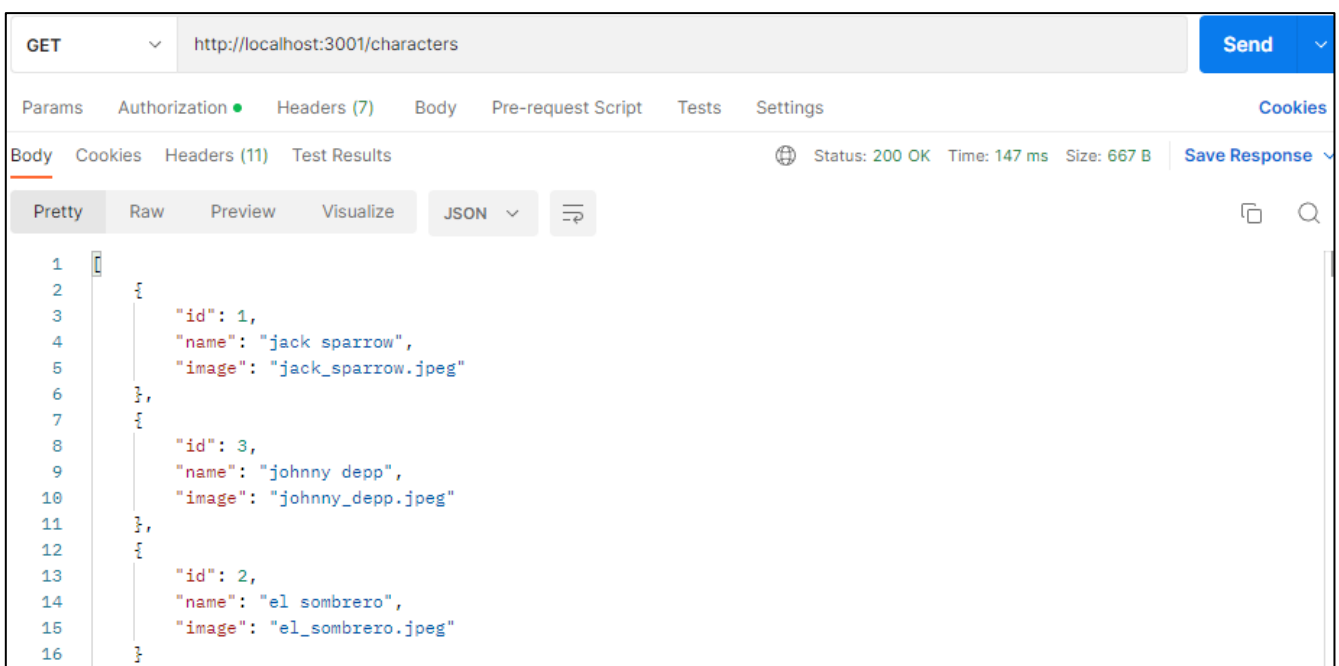
Permite crear permite crear personajes.

Recibe por body 'name', 'image', 'years', 'weight', 'history' y 'movies_series'. Comprueba que no se repita el nombre ingresado y que exista la película o serie.



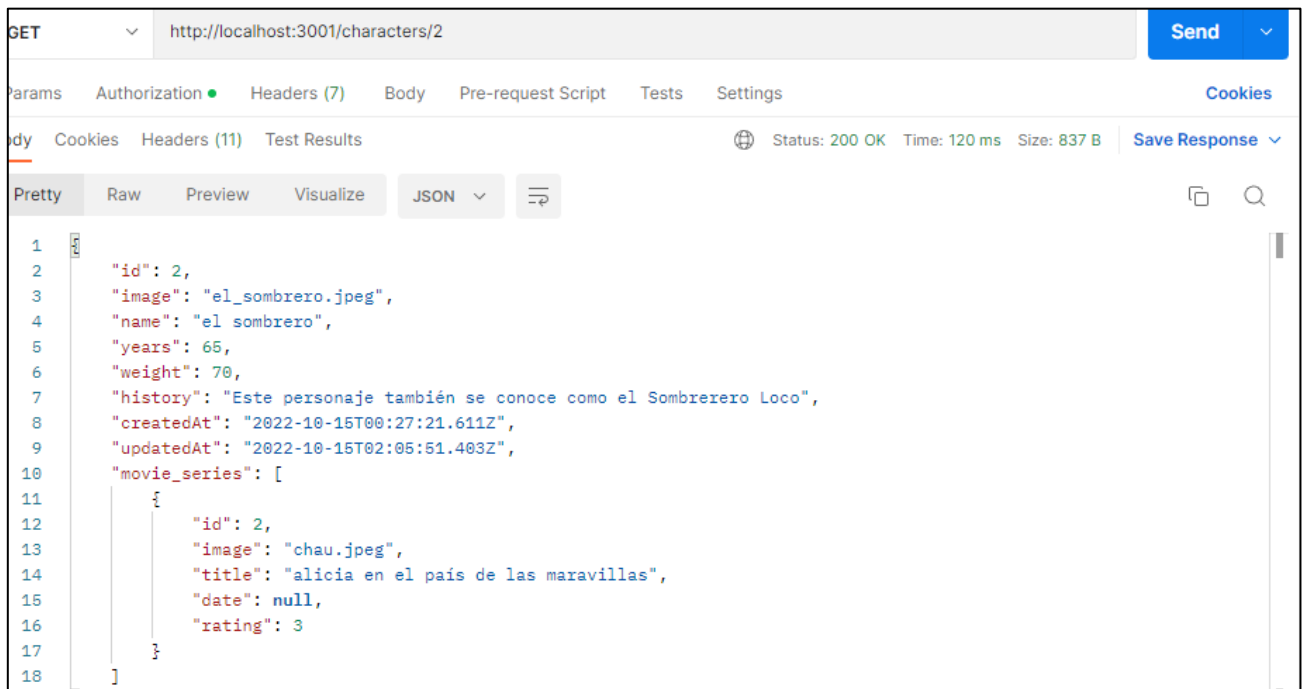
2- GET - '/characters'.

Permite crear listar personajes solo con los atributos 'name' e 'image'.



3- GET - '/characters/:id'.

Permite mostrar el detalle del personaje, cuyo 'id' es requerido por params. Muestra todos los atributos incluyendo las relaciones con la tabla 'Movie_serie'.



```
GET http://localhost:3001/characters/2
Status: 200 OK Time: 120 ms Size: 837 B
Save Response

Pretty Raw Preview Visualize JSON
1  {
2    "id": 2,
3    "image": "el_sombrero.jpeg",
4    "name": "el sombrero",
5    "years": 65,
6    "weight": 70,
7    "history": "Este personaje también se conoce como el Sombrero Loco",
8    "createdAt": "2022-10-15T00:27:21.611Z",
9    "updatedAt": "2022-10-15T02:05:51.403Z",
10   "movie_series": [
11     {
12       "id": 2,
13       "image": "chau.jpeg",
14       "title": "alicia en el pais de las maravillas",
15       "date": null,
16       "rating": 3
17     }
18   ]
}
```

4- GET - '/characters?querys=query's'.

Busca por nombre, filtra por años, por id de películas y por peso. Los parámetros son recibidos por query y son combinables, en caso de no recibir parámetros trae la lista de characters.

GET <http://localhost:3001/characters?name=jo&movies=1&weight=70> **Send**

Params • Authorization • Headers (7) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	jo	
<input type="checkbox"/> age	65	
<input checked="" type="checkbox"/> movies	1	
<input checked="" type="checkbox"/> weight	70	
Key	Value	Description

body Cookies Headers (11) Test Results **Status: 200 OK** Time: 18 ms Size: 618 B **Save Response**

Pretty Raw Preview Visualize JSON **JSON**

```

1  {
2    "id": 3,
3    "name": "johnny depp",
4    "image": "johnny_depp.jpeg",
5    "weight": 70,
6    "movie_series": [
7      {
8        "id": 1,
9        "title": "piratas del caribe"
10       }
11     ]
12   }

```

5- PUT - '/updatecharacter/:id'.

Permite editar Characters.

Recibe por body los datos que desea modificar, como por ejemplo 'name' y un 'id' por params.

PUT <http://localhost:3001/updatecharacter/1> **Send**

Params Authorization • Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded **raw** binary GraphQL **JSON** Beautify

```

1  {
2    "name": "jack sparrow",
3    "image": "jack_sparrow.jpeg",
4    "years": 59,
5    "weight": 72,
6    "history": "capitan del perla negra",
7    "movies_series": ["piratas del caribe"]
8  }

```

Body Cookies Headers (11) Test Results **Status: 200 OK** Time: 193 ms Size: 527 B **Save Response**

Pretty Raw Preview Visualize JSON **JSON**

```

1  "Character was updated successfully"

```

6- DELETE - '/deletecharacter/:id'.

Permite borrar Characters.

Recibe un 'id' por params.

DELETE

http://localhost:3001/deletecharacter/4

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Body

Cookies

Headers (11)

Test Results

Status: 200 OK

Time: 171 ms

Size: 520 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

"The character was deleted!"