

Universidad ORT Uruguay

Facultad de ingeniería

Diseño de aplicaciones 2

Obligatorio 1

<https://github.com/ORT-DA2/SobralGoni>

210361 - José Pablo Goñi

192247 - Juan Pablo Sobral

Criterios seguidos	3
Mecanismo de autenticación	4
Códigos de error	¡Error! Marcador no definido.
Descripción de los recursos	¡Error! Marcador no definido.

Colección de pruebas para todas las funcionalidades

Hay un archivo en el repositorio, llamado *DemoTests.postman_collection* en el cual van a poder encontrar pruebas de todas las funcionalidades pedidas.

Es también la colección utilizada a la hora de filmar el video para proveer evidencia de la ejecución exitosa de las mismas.

El video se puede encontrar en el siguiente link:

[https://www.youtube.com/watch?v= lihXsibgO8](https://www.youtube.com/watch?v=lihXsibgO8)

Documentación de casos de prueba para las funcionalidades prioritarias

En el repositorio tambien dejamos disponible una colección de pruebas de postman en formato archivo llamado PriorityTests.postman_collection.

A continuación vamos a entrar en detalle de las pruebas ejecutadas para cada una de las funcionalidades prioritarias.

El sistema utilizado para las pruebas posee unicamente un usuario administrador.

Previo al inicio de las mismas, se agregan 6 archivos(archivo1, archivo2, ...) y 2 carpetas(carpet1, carpeta2) en la carpeta raíz del usuario escritor. Luego se agrega un archivo dentro de cada una de ellas.

Mantenimiento de carpetas

- Prueba obtener carpeta para verificar que tiene un nombre y una referencia a la carpeta en la que se encuentra.

The screenshot shows a Postman interface with a POST request to `http://localhost:57901/api/users/`. The request body is a JSON object representing a user. The response is a detailed JSON object showing the user's information and a newly created folder.

```
1 {
2   "firstName": "User",
3   "lastName": "I",
4   "username": "user1",
5   "password": "user1",
6   "email": "user@user.user",
7   "Role": "User"
8 }
```

Response (JSON):

```
1 {
2   "id": 4,
3   "firstName": "User",
4   "lastName": "I",
5   "username": "user1",
6   "password": "user1",
7   "email": "user@user.user",
8   "role": "User",
9   "token": null,
10  "friendList": [],
11  "rootFolder": {
12    "files": [],
13    "folders": [],
14    "id": 1,
15    "ownerId": 4,
16    "name": "user1-rootFolder",
17    "parent": null,
18    "readers": []
19  }
20 }
```

Como se puede ver, cuando se crea un usuario tambien se crea una carpeta, en este caso se creo al usuario user1, y consiguientemente su carpeta raiz, de nombre user1-rootFolder, id 1 y parent null.

Ahora hago un GET de esa carpeta y recibo lo siguiente:

The screenshot shows a REST client interface with a tab for 'GET GetFolder'. The URL is 'http://localhost:57901/api/folders/1'. The 'TYPE' is set to 'Bearer Token'. The 'Token' field contains a long alphanumeric string. The 'Send' button is highlighted. Below the request, the 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a 200 OK status with a time of 113ms and a size of 426 B. The JSON body contains an array of folders, with the first folder having an id of 1 and a name of 'user1-rootFolder'. The 'readers' array contains one user with id 4, first name 'User', last name '1', username 'user1', password 'user1', email 'user@user.user', role 'User', token null, and friendList null.

```
1 {
2   "files": [],
3   "folders": [
4     {
5       "id": 1,
6       "ownerId": 4,
7       "name": "user1-rootFolder",
8       "parent": null,
9       "readers": [
10        {
11          "id": 4,
12          "firstName": "User",
13          "lastName": "1",
14          "username": "user1",
15          "password": "user1",
16          "email": "user@user.user",
17          "role": "User",
18          "token": null,
19          "friendList": null
20        }
21      ]
22    }
23  ]
24 }
```

El resultado es el esperado.

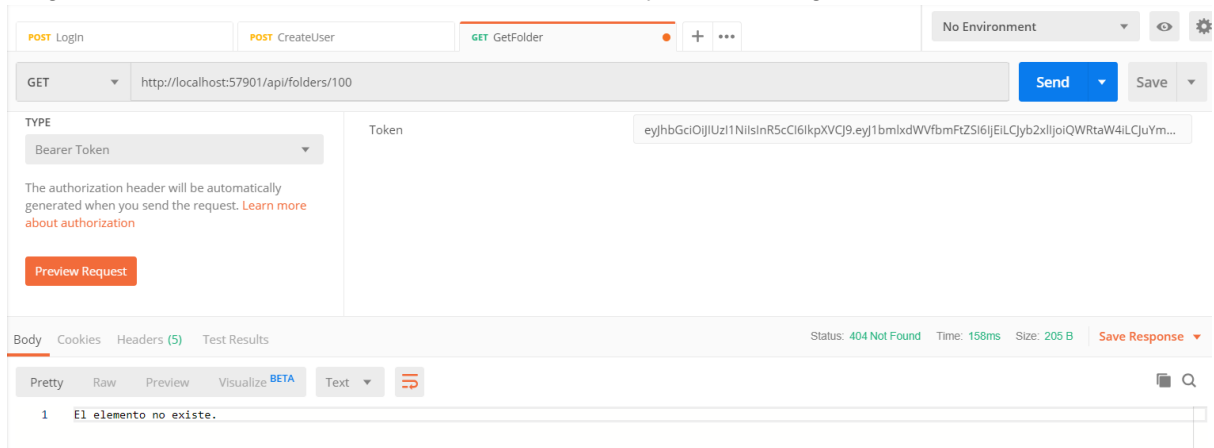
Hago un GET a una ruta invalida (/folders/pepe) y recibo lo siguiente:

The screenshot shows a REST client interface with a tab for 'GET GetFolder'. The URL is 'http://localhost:57901/api/folders/pepe'. The 'TYPE' is set to 'Bearer Token'. The 'Token' field contains a long alphanumeric string. The 'Send' button is highlighted. Below the request, the 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a 400 Bad Request status with a time of 84ms and a size of 369 B. The JSON body contains an error message: 'The value 'pepe' is not valid.' The 'title' is 'One or more validation errors occurred.', the 'status' is 400, and the 'traceId' is '8000001c-0001-fb00-b63f-84710c7967bb'.

```
1 {
2   "errors": {
3     "folderId": [
4       {
5         "The value 'pepe' is not valid."
6       }
7     ],
8     "title": "One or more validation errors occurred.",
9     "status": 400,
10    "traceId": "8000001c-0001-fb00-b63f-84710c7967bb"
11  }
```

El resultado es el esperado.

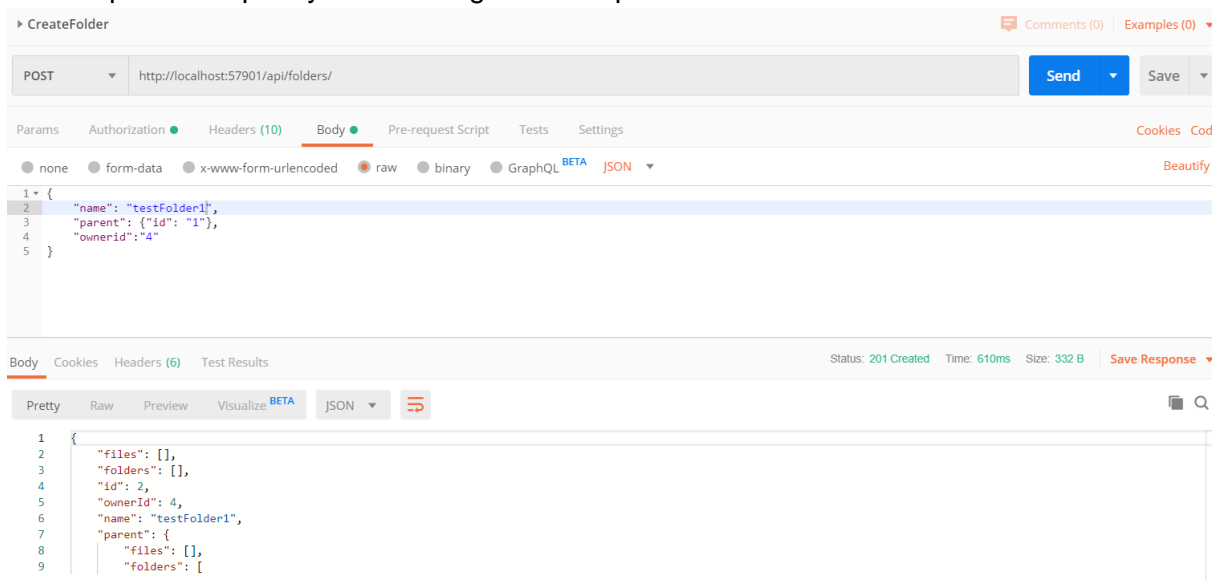
Hago un GET a una ruta de un folder inexistente y recibo lo siguiente:



Código de error 404, el elemento no existe, respuesta esperada.

- Prueba crear dos carpetas con el mismo nombre dentro de la misma carpeta:

Creo la primer carpeta y recibo la siguiente respuesta:



Intento crear otra carpeta con el mismo nombre y recibo la siguiente respuesta:

The screenshot shows a REST client interface for a POST request to `http://localhost:57901/api/folders/`. The request body is a JSON object: `{ "name": "testFolder1", "parent": { "id": "1" }, "ownerId": "4" }`. The response status is `404 Not Found` with the message `El nombre de la carpeta ya existe.` (The folder name already exists).

Indicando el correcto funcionamiento del sistema

- Prueba borrar una carpeta y consiguientemente todo su contenido

Primero que nada creo un archivo dentro de la carpeta recién creada:

The screenshot shows a REST client interface for a POST request to `http://localhost:57901/api/files/`. The request body is a JSON object: `{ "name": "fileName", "parent": { "id": "2" }, "ownerId": "4", "content": "fileContent" }`. The response status is `201 Created` with a detailed JSON object: `{ "content": "fileContent", "creationDate": "2019-10-08T19:08:39.0771526-03:00", "lastModifiedDate": "2019-10-08T19:08:39.0768653-03:00", "id": 1, "ownerId": 4, "name": "fileName", "parent": { "files": [...] } }`.

La respuesta es correcta, el id del archivo es uno.

Pruebo acceder al archivo:

CreateFile Copy Comments (0) Examples (0)

GET http://localhost:57901/api/files/1 Send Save

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Code

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL BETA

This request does not have a body

Body Cookies Headers (5) Test Results Status: 200 OK Time: 130ms Size: 359 B Save Response

Pretty Raw Preview Visualize BETA JSON ⌵

```
1 {
2   "content": "file1Content",
3   "creationDate": "2019-10-08T19:08:39.0771526",
4   "lastModifiedDate": "2019-10-08T19:08:39.0768653",
5   "id": 1,
6   "ownerId": 4,
7   "name": "file1Name",
8   "parent": {
9     "files": [
```

Accedo sin problemas.

Pruebo borrar la carpeta(id 2):

DeleteFolder Comments (0) Examples (0)

DELETE http://localhost:57901/api/folders/2 Send Save

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

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL BETA

This request does not have a body

Body Cookies Headers (3) Test Results Status: 204 No Content Time: 148ms Size: 115 B Save Response

Pretty Raw Preview Visualize BETA Text ⌵

```
1 
```

El codigo 204 indica que la carpeta fue borrada de manera correcta.

Pruebo acceder nuevamente al archivo que estaba dentro de la carpeta:

CreateFile Copy Comments (0) Examples (0)

GET http://localhost:57901/api/files/1 Send Save

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Code

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL BETA

This request does not have a body

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 91ms Size: 205 B Save Response

Pretty Raw Preview Visualize BETA Text ⌵

```
1 El elemento no existe.
```

La respuesta es un 404, correcta.

Se permite mover una carpeta a otra, moviendo con ella todo su contenido. No se puede mover una carpeta a una de sus hijas.

Primero que nada creo dos carpetas:

The image displays two screenshots of a REST client interface, likely Postman, showing the process of creating two folders via a POST request to the endpoint `http://localhost:57901/api/folders/`.

Top Screenshot: The request body is a JSON object:

```
{  "name": "testFolder1",  "parent": {"id": "1"},  "ownerId": "4"}
```

. The response status is **201 Created** with a time of 318ms and a size of 332 B. The response body, shown in JSON format, is:

```
{  "files": [],  "folders": [],  "id": 2,  "ownerId": 4,  "name": "testFolder1",  "parent": {    "files": [],    "folders": []  }}
```

Bottom Screenshot: The request body is a JSON object:

```
{  "name": "testFolder2",  "parent": {"id": "1"},  "ownerId": "4"}
```

. The response status is **201 Created** with a time of 198ms and a size of 400 B. The response body, shown in JSON format, is:

```
{  "files": [],  "folders": [],  "id": 3,  "ownerId": 4,  "name": "testFolder2",  "parent": {    "files": [],    "folders": []  }}
```

La respuesta es correcta, se puede ver que se crean ambas carpetas.

Pruebo crear un archivo en la carpeta 2:

The screenshot shows a REST client interface for a POST request to `http://localhost:57901/api/files/`. The request body is a JSON object:

```
{  "name": "fileName",  "parent": { "id": "2" },  "ownerId": "4",  "content": "fileContent"}
```

. The response status is 201 Created, with a time of 246ms and a size of 421 B. The response body is a JSON object:

```
{  "content": "fileContent",  "creationDate": "2019-10-09T17:24:03.7189275-03:00",  "lastModifiedDate": "2019-10-09T17:24:03.7186432-03:00",  "id": 1,  "ownerId": 4,  "name": "fileName",  "parent": {    "files": [
```

La respuesta es correcta, se puede ver que se la crea dentro de la carpeta con el id 3.

Pruebo mover la carpeta del id 2 a la carpeta del id 3:

The screenshot shows a REST client interface for a PUT request to `http://localhost:57901/api/folders/2/folder/3`. The request is authenticated with a Bearer Token: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmtdWVfZmFZS6jQILCjyb2xlljoVXNlcislm5iZiI6M...`. The response status is 200 OK, with a time of 160ms and a size of 206 B. The response body is a plain text message: `The folder was moved correctly`.

El resultado es correcto.

Realizo un GET sobre la carpeta 3, la cual ahora debería tener dentro de sí a la carpeta 2

POST Login GET GetFolder POST CreateFol... POST CreateFile GET GetFile PUT MoveFolder POST CreateUser + ...

GetFolder

GET http://localhost:57901/api/folders/3

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

TYPE
Bearer Token

Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmlxdWVfb

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Body Cookies Headers (5) Test Results Status: 200 OK

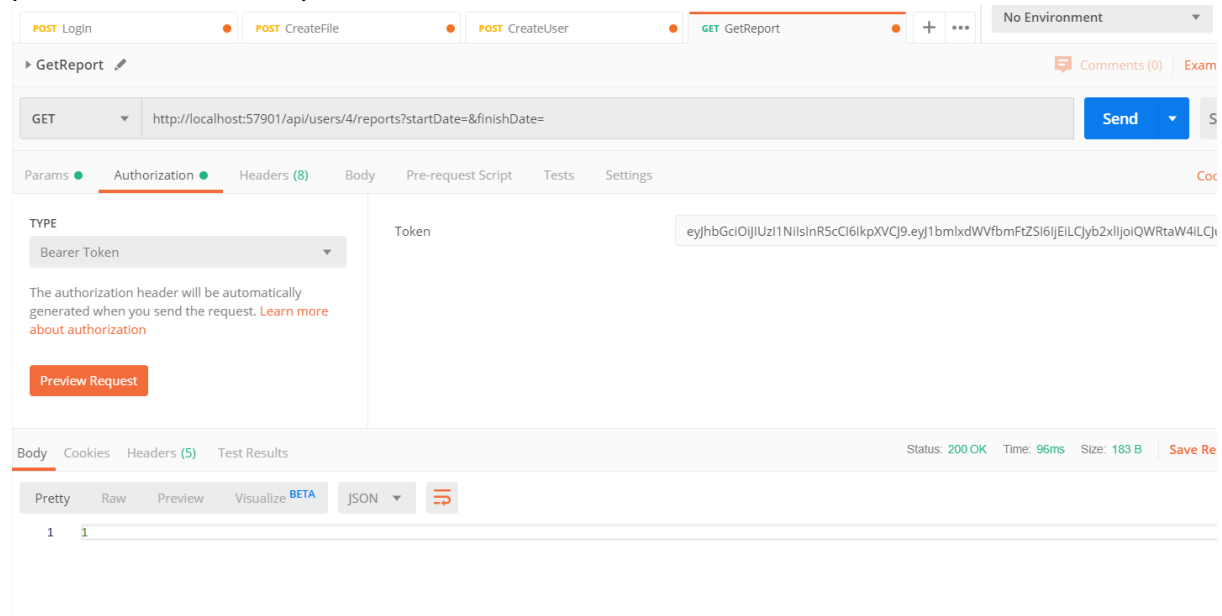
Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "files": [],
3   "folders": [
4     {
5       "files": [
6         {
7           "content": "file1Content",
8           "creationDate": "2019-10-09T17:24:03.7189275",
9           "lastModifiedDate": "2019-10-09T17:24:03.7186432",
10          "id": 1,
11          "ownerId": 4,
12          "name": "fileName"
```

La respuesta es correcta. Se ve que se movio la carpeta, y dentro de la misma esta el archivo.

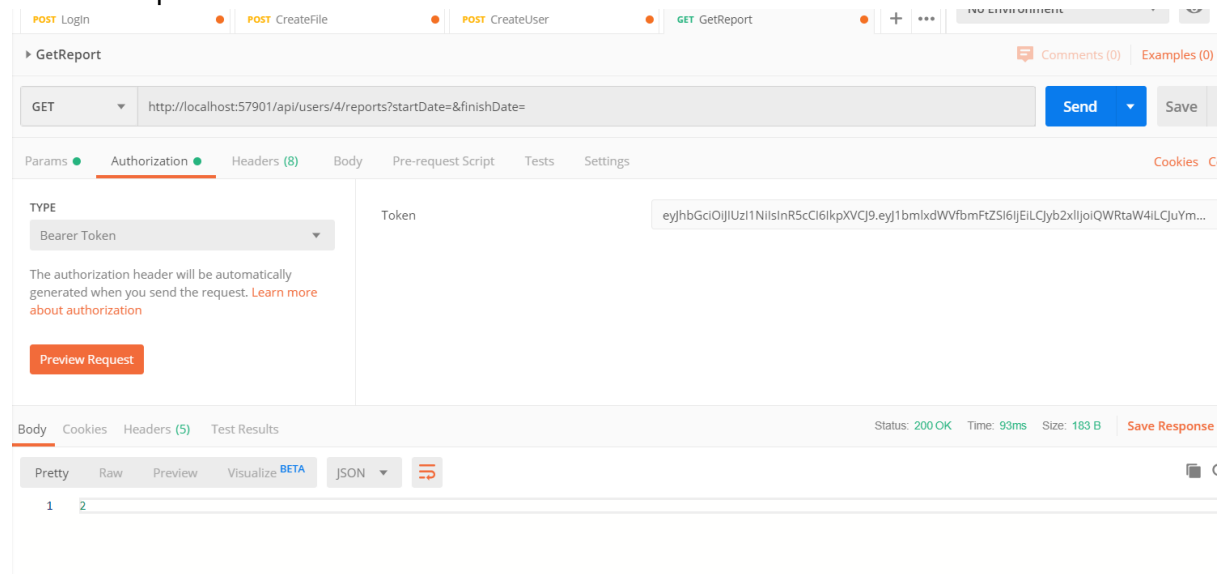
Cantidad de modificaciones sobre archivos (nuevo, edición, borrado), llevada a cabo por un usuario por día en determinado periodo de tiempo (entre dos fechas).

Para esta prueba se limpia la BD, se crea un usuario nuevo el cual crea siete archivos en su raíz. Luego llamamos al reporte sin pasarle parámetros, dado que en ese caso calcula el promedio de archivos para la última semana, es decir, 7 días:



The screenshot shows the Postman interface for a GET request to `http://localhost:57901/api/users/4/reports?startDate=&finishDate=`. The request is configured with a Bearer Token. The response status is 200 OK, and the response body is a JSON array containing the number 1.

El resultado es 1, que es el esperado, ahora agrego archivos hasta llegar a 14 y vuelvo a llamar al reporte:



The screenshot shows the Postman interface for a GET request to `http://localhost:57901/api/users/4/reports?startDate=&finishDate=`. The request is configured with a Bearer Token. The response status is 200 OK, and the response body is a JSON array containing the number 2.

El resultado es 2, que es el esperado.