

# Peer review of A3 for lab 183

By lab 181 - Lodewijk & Caio

## Documentation

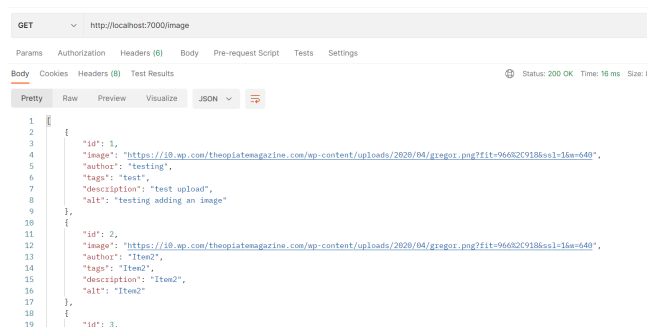
Overall, the quality of the documentation was great and consistent for all endpoints. They make it very clear what has to be included to make a successful request and what are the endpoints and methods that should be used, not only that but examples are included to show successful and unsuccessful requests. The tables of all parameters that could be included was a very nice touch, showing us explicitly what can be included and what is required in an easy to read and find way.

Name	Required?	Data Type	Description
id	yes	integer	image id
author	yes	string	image author
image	yes	string	image url
tags	yes	string	image tags
alt	yes	string	short image description
description	yes	string	image description

Table of all possible parameters in the create request. Clearly indicating there name and if they are required

## Retrieving the full dataset

By using the endpoint url stated in the documentation I was able to easily retrieve the entire database, returned as a json object. Even when changing the database this method always returns the full database. Interestingly, the function to retrieve a single item is the same as the one to retrieve the full dataset, allowing them to save many lines as these functions are very similar.



```
GET http://localhost:7000/image

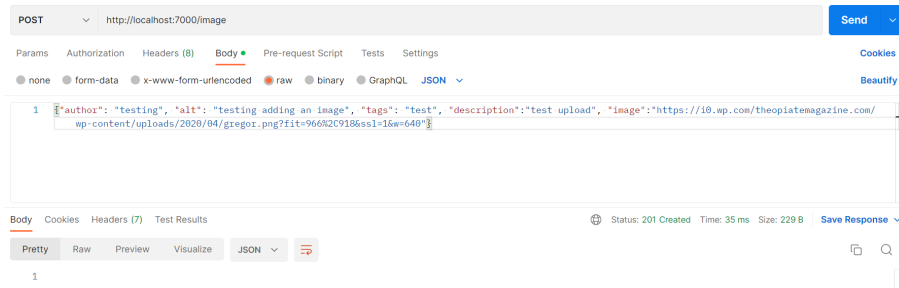
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (6) Test Results Status: 200 OK Time: 16 ms Size: 858

Pretty Raw Preview Visualize JSON

1 {
2   {
3     "id": 1,
4     "image": "https://lh.ap.com/theplatemagazine.com/wp-content/uploads/2020/04/gregor.png?fit=966&C918&ssl=1&w=640",
5     "author": "testing",
6     "tags": "test",
7     "description": "test upload",
8     "alt": "testing adding an image"
9   },
10  {
11    "id": 2,
12    "image": "https://lh.ap.com/theplatemagazine.com/wp-content/uploads/2020/04/gregor.png?fit=966&C918&ssl=1&w=640",
13    "author": "Item2",
14    "tags": "Item2",
15    "description": "Item2",
16    "alt": "Item2"
17  },
18  {
19    "id": 3,
```

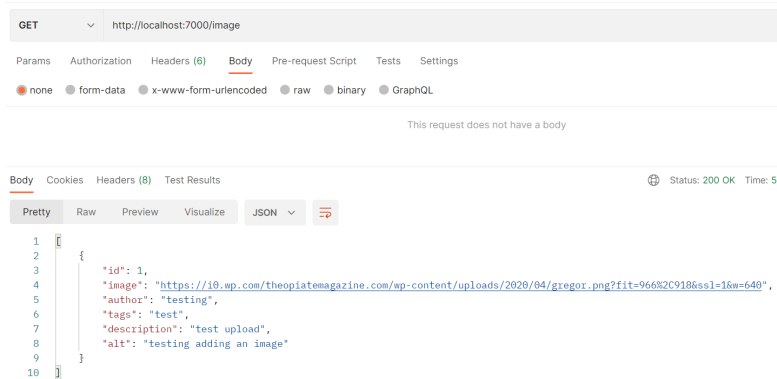
Using get function with no id specified returns all objects with a 200 status code

## Adding data (Create)



Running post with the specified endpoint returns nothing and a 201 status code

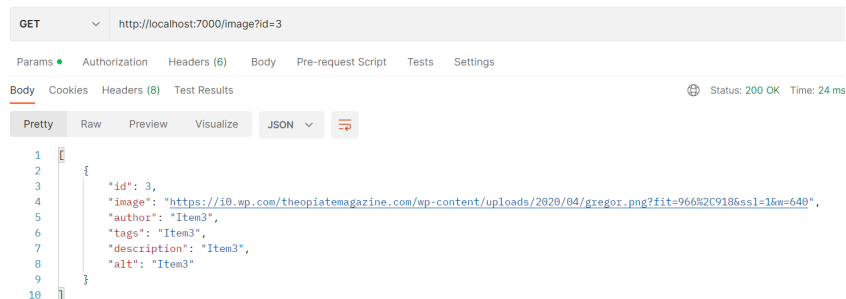
The API is able to create and add new data to the database, returning a 201 status code. The error messages that are possible when one of the fields are missing are very interesting, as it will change depending on what field has not been filled in properly.



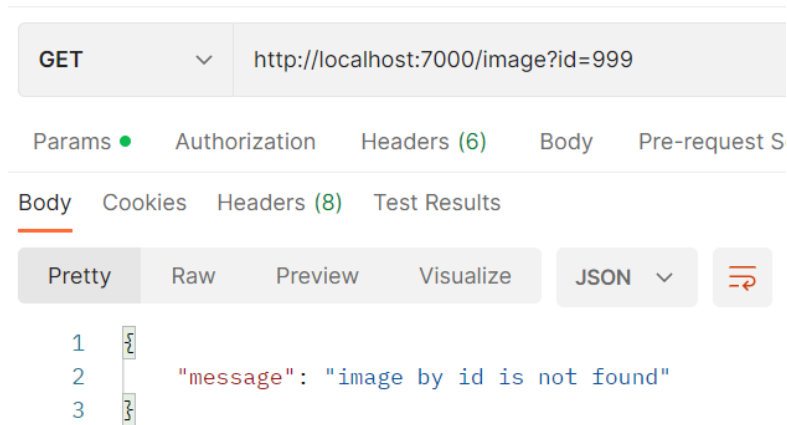
What the server get function returns after running post, thus the item was added to the database

## Listing data for a specific item (Retrieve)

Retrieving a specific item uses the same function as retrieving all items but with an id parameter. The error messages used account for when the id is invalid, such as when it is a letter or word instead of a number, and also when the id does not exist in the database.



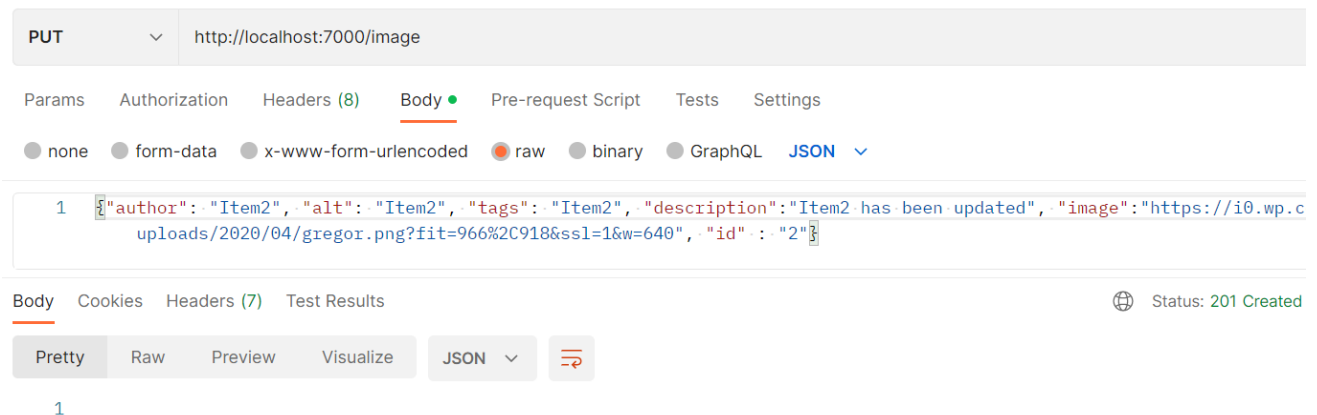
Specifying an id on the get function returns only a specific item and a 200 status code



If the id doesn't exist in the database a 404 status and the following message are returned

## Changing data of a specific item (Update)

Using the API you are able to update existing table elements, however, it also allows you to update elements that also aren't included in the database making its use intersect with that of the adding data functionality. Instead of allowing this it should instead cause an error saying that this id is invalid in the current database. The rest of the error handling does account for when the id imputed isn't a number, but does not for when the id is a decimal number or a negative number which shouldn't be possible. Although trying to add a decimal number returns successfully but doesn't add it to the database.



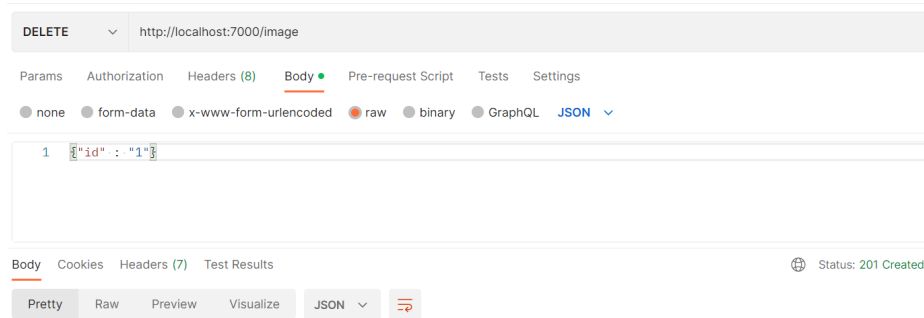
When post runs successfully it updates a database item and returns a 201 status code

```
[
  {
    "id": -1,
    "image": "https://i0.wp.com/theopiatemagazine.com/wp-content/uploads/2020/04/gregor.png?fit=966%2C918&ssl=1&w=640",
    "author": "Item-1",
    "tags": "Item-1",
    "description": "testing",
    "alt": "Item-1"
  }
]
```

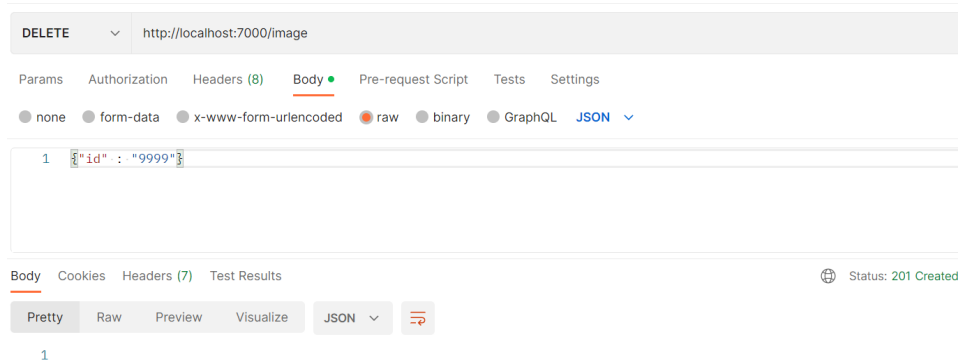
Using post I added an item with a -1 id

## Removing a specific item (Delete)

Even though the api can properly delete items from the database, the status code returned, 201, differs from the one specified in the documentation, 204. Not only that, but it is improper for this case, as it is normally used when an item is created. In terms of error handling, it once again accounts for when the id is not a number but not when the id is not present in the database or when it's a decimal number.



Using delete removes an item from the database and returns a 201 status code



Trying delete with an invalid id returns successfully when it shouldn't.

## Has the bonus been implemented? If so, does the bonus work correctly?

The bonus has been implemented into the website and all CRUD functionality works except for the deletion of items, which is only possible on the topmost image of the table. Not only that, but all previous website features such as the search bar and search filters still work perfectly.

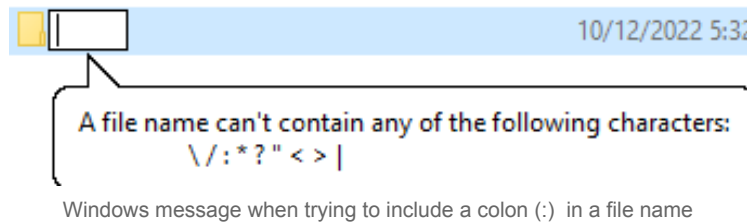
```
const API = "http://127.0.0.1:7000/image";
```

Code from there javascript shows they are using their own api as the API is set to the local host

## Actionable feedback

Although this code is fairly good, it has two major issues. Firstly, in order for me to be able to review this code I had to first debug it, because due to the file name for the database it was unusable on my Windows computer. This was caused by the fact the database name starts with

a colon (:), a character which Windows restricts from being in file names. Next time be careful with the name choices for your files, as using characters that might be restricted on certain operating systems limits the possible users for your api even though it would work otherwise. Secondly, remember to cast a wide net for possible errors, as many of your functions allow clients to make requests that aren't necessarily valid but are still returning a successful status code. Also make sure to go through your documentation and implementation to make sure that the status codes supposed to be returned are the same.



## What we learnt / weren't aware of

The implementation of many features, such as the conjoining of the retrieve a specific item and retrieve all items function, was quite interesting and provided insight on not only how to make our code shorter and more effective but also shorter and clearer. The error handling messages were also a learning opportunity as instead of making one general one, they were specific and told clients exactly what caused the error instead of a general area that could have caused the error.