Rest api Introduction

Introduction

**What is a REST API ?**

Let’s say you’re trying to find Batman videos on Youtube. You open up Youtube, type “Batman” into the search field, hit enter, and you see a list of videos about Batman. A REST API works in a similar way. You search for something and you get a list of results back from the service you’re requesting it from.

**API: Application Programming Interface**

It is a set of rules that allows programs to talk to each other. The developer creates the API on the server and allows the client to talk to it.

**REST: REpresentational State Transfer**

Determines how the API looks like. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL

=======

**What does REST stand for ?**

REpresentational State Transfer

REpublish State Transfer

REpresentational State Transition

====

**RESTful is a set of recommendations.**

True

False

====

**RESTful API is only an API that is created under Node.js**

True

False

**Request**

The internet boasts a vast array of resources hosted on different servers. For you to access these resources, your browser needs to be able to send a request to the servers and display the resources for you. HTTP (Hypertext Transfer Protocol) is the underlying format that is used to structure requests and responses for effective client-server communication. The message that is sent by a client to a server is what is known as an HTTP request. Clients can use various methods to send requests to a server.

Therefore, HTTP request methods are the assets that indicate the specific desired action to be performed on a given resource. Each method implements a distinct semantic, but there are some standard features that are shared by the various HTTP request methods.

To test HTTP requests, we can use softwares like Postman

The request is sent from the browser to a server.

True

False

The request is sent via the SMTP protocol.

True

False

The HTTP request can only have one type of method.

True

False

Request Headers

Request Headers

Headers are used to provide information to both the client and the server. It can be used for many purposes, such as authentication and providing information about the body content.

Example of a HTTP Request Headers:



Request headers are used to hold messages.

True

False

Request headers provide information to servers and clients.

True

False

The request header holds only the authentication information.

True

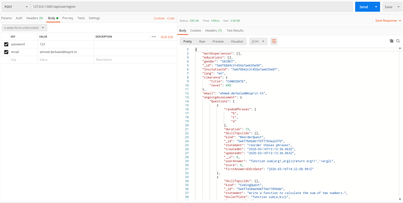
False

Requeqt Body

Request Body

The body (sometimes called “data” or “message”) contains information you want to be sent to the server. This option is only used with POST, PUT or DELETE requests.

For a sign in request, a body example can be the email and password.



The request body can also be called

Data

Message

Statement

The request body is used with the get method

True

False

The request body contains the result of the request.

True

False

Response

HTTP Response

When a client (browser) makes a request to a server, the server responds with a reply including a status code.

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

Informational responses (100–199),

Successful responses (200–299),

Redirects (300–399),

Client errors (400–499),

and Server errors (500–599).



The response of a successful POST 127.0.0.1:30001/product/create request will have a status code equal to?

101

201

301

401

501

The response status varies between 100 and 599.

True

False

The response is sent from the client to the server.

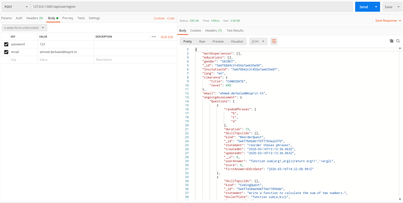
True

False

Post( REST API)

POST Method

This request is used to create a new resource on a server. If you perform a POST request, the server creates a new entry in the database and tells you whether the creation is successful or not. In other words, a POST request performs an CREATE operation.

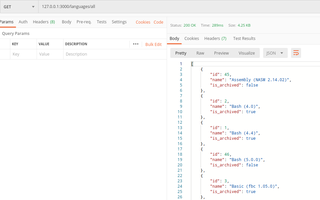


We use POST requests to make CREATE operations. The best suitable Response Status code for similar requests is 201.

Get (Rest API)

GET Method

This method is used to get a resource from a server. If you perform a GET request, the server looks for the data you requested and sends it back to you. In other words, a GET request performs a READ operation. This is the default request method.



The get method is used to send data from the client to the server.

True

False

The get method performs a read operation.

True

False

The get method is an HTTP method.

True

False

PUT Method

This request is used to update a resource on a server. If you perform a PUT request, the server updates an entry in the database and tells you whether the update is successful or not. In other words, a PUT request performs an UPDATE operation.

When sending a PUT request, we add a body containing the modification to be performed.

The PUT method is used for the update operation.

True

False

The update method contains a body.

True

False

The PUT method performs an update on a database entity.

True

False

DELETE Method

This request is used to delete a resource from a server. If you perform a DELETE request, the server deletes an entry in the database and tells you whether the deletion is successful or not. In other words, a DELETE request performs a DELETE operation.

To delete a document, we send a PUT request.

True

False

The delete method is not a HTTP request, it has to be implemented.

True

False

After the delete method is performed, the server sends back a success or failure response.

True

False

Conclusion

Recap

We've learned what a REST API is and how to use POSTMAN to perform a request with GET, POST, PUT, and DELETE methods. Next, we will learn how to interact with a REST API in a Node JS application.

As every superskill, we prepared a cheatsheet with everything you learned in this chapter:

📜 GMC REST API Cheat Sheet

Checkpoint

**Checkpoint Objective**

In this checkpoint, you will be performing a series of guided instructions to create a REST API by manipulating your database with mongoose.

* PS: don't forget to comment on your code before submitting it.

**Instructions**

1. Start a new Node JS project  with ‘ npm init ‘
2. Install the mongoose and express and  .env
3. Configure the environment variables with .env
4. Lunch a server with express in the server.js file
5. Connect your database locally or with mongo atlas
6. The Folder structure should be like this :

config/ .env

Server.js

     7. Create a models folder and a User.js file in it

     8. In User.js you must define a mongoose Schema and export the model , you will use it in the server.js

     9. The Folder structure should be like this :

config/ .env

models/User.js

Server.js

     10. In the server.js create four routes :

       GET :  RETURN ALL USERS

       POST :  ADD A NEW USER TO THE DATABASE

       PUT : EDIT A USER BY ID

       DELETE : REMOVE A USER BY ID

**NB :** in each callback function you will use mongoose methods to manipulate your data and return it in the response

     11.  Use postman to test each route.

**Useful Links :**

1. .env : <https://www.npmjs.com/package/dotenv>
2. Express js : <https://expressjs.com/>
3. Req.params && req.query : <https://coursework.vschool.io/express-params-and-query/>
4. Mongoose : <https://mongoosejs.com/>
5. Rest api : <https://www.youtube.com/watch?v=SLwpqD8n3d0>