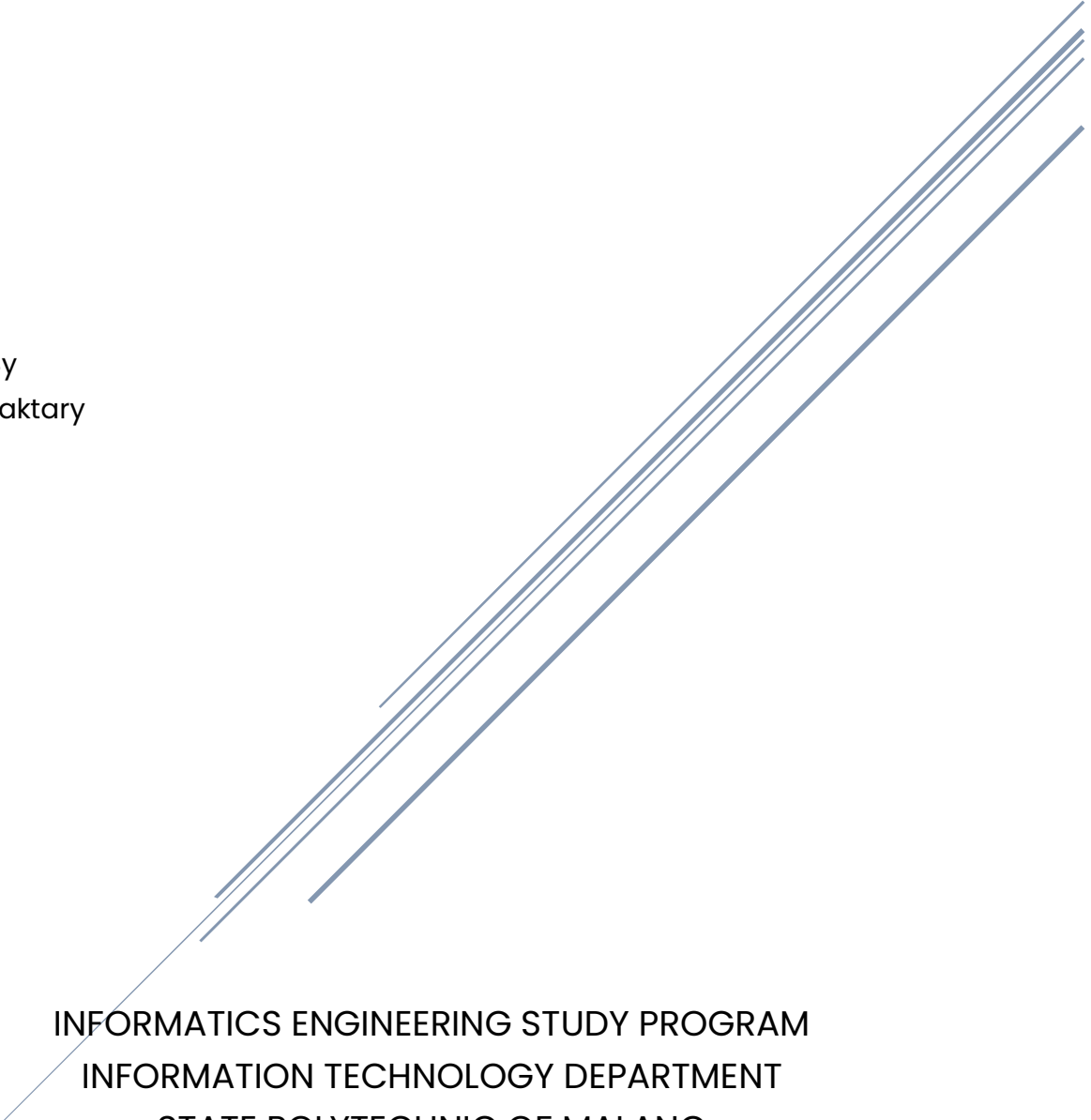# GUIDE A05

Create a DELETE Endpoint to Delete Data

Arranged By
Omar Al-Maktary

INFORMATICS ENGINEERING STUDY PROGRAM
INFORMATION TECHNOLOGY DEPARTMENT
STATE POLYTECHNIC OF MALANG
2023

Contents

Create a DELETE Endpoint to Delete Data

Objectives

1. Students understand how to create a DELETE endpoint to delete a product resource to the database using a slug as the product identifier.
2. Students understand how to delete a product resource from the web interface of the project.

Learning the method DELETE and how to create an endpoint to delete an existing resource in a MongoDB collection are the goals of guide A05. The practice of this guide will be used in same application established in guides A01, A02, A03, and A04.

In this material, Students are expected to learn how to create a method that will handle requests for deleting product resources from the web interface of the project that has been created in the previous guides.

Requirements

Having the correct hardware and software components is essential for ensuring the successful execution of the tasks outlined in this guide. The hardware configuration and software required for completing this guide tasks are as the following:

Hardware Specifications

The minimum hardware specifications for running a Node.js API application on the Windows operating system and using software such as Postman and Visual Studio Code are the following:

Minimum Requirements

- Processor: Intel Core i3 or equivalent.
- RAM: 4 GB.
- Storage: 500 GB HDD with at least 20 GB of available storage.
- Graphics: Integrated graphics card.
- Connectivity: Ethernet and Wi-Fi capabilities.

Recommended Requirements

- Processor: Intel Core i5 or equivalent.
- RAM: 8 GB or more.
- Storage: 256 GB SSD with at least 20 GB of available storage.
- Graphics: Integrated graphics card.
- Connectivity: Ethernet and Wi-Fi capabilities.

Software required

It is important to have the correct software installed on your system to ensure that the application runs smoothly and meets performance expectations. The software required is as follows:

- Operating System: Windows 10 or later.
- NodeJS: Latest stable version installed.
- Visual Studio Code: Latest stable version installed.
- Postman: Latest stable version is installed.

Note: NodeJS, Visual Studio Code, and Postman installation have been explained in the previous guide, A01.

NPM Packages

- nodemon: Automatically restarts Node application on file changes.
- cross-env: Sets environment variables in a cross-platform way.
- jest: Creates and executes tests.
- jest-expect-message: Enhances Jest assertions with custom error messages.
- jest-image-snapshot: Adds image snapshot testing to Jest.
- puppeteer: Node library to control a headless Chrome or Chromium browser.
- supertest: Makes HTTP queries to the application and checks results.
- dotenv: Simplifies management of environment variables.
- express: NodeJS framework for creating apps with routing and middleware.
- ejs: Embedded JavaScript templating.
- express-ejs-layouts: Layout support for EJS in Express.
- mongoose: MongoDB object modeling library for NodeJS.
- mongoose-slug-generator: Automatically generates slugs based on a Mongoose schema field.

Resource

- Documents: Guide A05
- Tests: api/testA05.test.js, web/testA05.test.js

Task Description

Students can create a DELETE endpoint to delete a product from the database. When a client sends a request to this endpoint, the function responsible for handling it should be able to receive the product's unique identifier (slug). If the product was not found the server should return an error informing that the product was not found. If the slug was found the product resource will be deleted. Students should also create a function that will handle any request from the web interface to delete a product resource.

Start Coding

1. Open the file "controllers/api/product.controller.js". and copy the following code to it:

```
const deleteProduct = async (req, res) => {
 try {
 } catch (error) {
   return res.status(500).json(error);
 }
};
```

Figure 1 deleteProduct Function Initial Code

Don't forget to export the function **deleteProduct** as the following:

```
module.exports = {
 getProducts,
 getProduct,
 createProduct,
 updateProduct,
 deleteProduct,
};
```

Figure 2 "controllers/api/product.controller.js" Exports Functions Code

2. Go to the file "routes/api/product.routes.js", and define a new route as the following:

```
router.delete("/product/:slug", deleteProduct);
```

Figure 3 "routes/api/product.routes.js" New Route

3

The function **deleteProduct** needs to be required from the controller to handle the requests to the DELETE endpoint "/api/v1/product/:slug" similar to the following code:

```javascript
const {
  getProducts,
  getProduct,
  createProduct,
  updateProduct,
  deleteProduct,
} = require("../controllers/product.controller");
```

Figure 4 product.controller.js Required Functions

3. Go back to the file "product.controller.js" and update the function **deleteProduct** to fit the requirements of the following table. This table contains information needed for the endpoint details.

DELETE "/api/v1/product/:slug" ENDPOINT STRUCTURE			
API Endpoint Path	*Request Method*	*Response Format*	*Description*
"/api/v1/product/:slug"	DELETE	JSON	Deletes an existing product.
Request Parameters			
Parameter	*Type*	*Description*	
"slug"	String	The slug of the product to delete.	
Response Parameters			
Parameter	*Type*	*Description*	
"product"	Object	An object containing information about the deleted product.	
"message"	String	A message indicating the status of the response.	
Success Responses			
HTTP Status Code	*Response*		
200	{ "product": product, "message": "Product deleted" }		
Error Responses			
HTTP Status Code	*Response*		
404	{ "message": "No product found" }		
500	{ error object }		

Table 1 Endpoint DELETE "/api/v1/product/:slug" API Architecture Design

Note that the **product** in the Success Responses section is an instance of the product object after being deleted.

4. Hints: use the following code to help in completing the task:

```
const product = await Product.findOneAndDelete({ slug: req.params.slug});
```
Figure 5 Delete a Product Code Hint

The **findOneAndDelete** method is provided by Mongoose to delete the product that matches the criteria in this case it is the slug value that will act as a more friendly identity for the object.

Run The API Application

To run the application, use the following command:

For this guide and development purposes the command "npm run dev" is used to execute the command "nodemon server.js" which will run the "server.js" using the nodemon package. This package allows the server to reload if any changes occur in the code of the application. Run the development command "npm run dev" in the terminal and notice the console message.

Testing The API Application

In this section, several tests in different ways will be explored to verify the results of the student's work on this document.

Using Postman

To verify results using Postman, follow these steps:

1. Create a new request in the folder "api-experiment" with the option DELETE as its method.
2. Use the following link as the URL.

    {{protocol}}{{host}}{{port}}{{version}}/product/:slug

    Change the ":slug", to an existing product slug in the database.
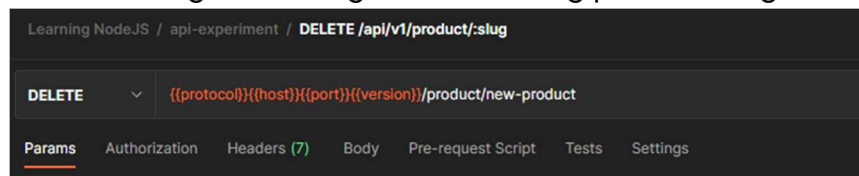
Figure 6 Postman Request URL

3. Send the request, it should return the deleted product object as a MongoDB document and a message indicating the product has been deleted.

Figure 7 Postman Response Results Deleted

Note that the status is 200 meaning the request is OK. Additionally, notice the message indicating the product data was deleted. This also can be verified by viewing the data on the MongoDB Atlas panel, it should be deleted.

4. Try to send the same request again, it should return an error that the product is not found.
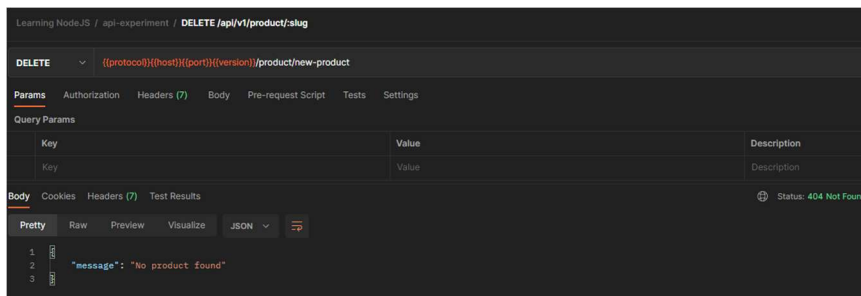
Figure 8 Postman Response Results Not Found

Note that the status is 404 indicating content not found. The response message shows that the product was not found.

Running The API Test file

Note: Sometimes the test will have an error of time limit, try to re-run the test or increase the testTimeout in the next step.

1. Copy the file "testA05.test.js" from the "api" folder within the "tests" folder for this material to the "tests/api" folder of your project base directory.
2. In the terminal run the command "npm run api-testA05" and notice the results.
3. If everything is correct and working well the results in the terminal should look as the following:

6

Figure 9 Successful Test Results

4. If the test failed and it shows an error similar to the following figure:

Figure 10 Failed Test Result

Try to find out why the test failed and fix it until the test result shows successful results.

Creating The Web Application

In this section, the web interface for this application will not have new pages added to it. The same basic endpoints explained in the previous sections will be implemented in a web page that can delete an existing product.

To start working on the web interface, follow these steps:

1. In the "controllers/web/product.controller.js" file, create a new function named "**deleteProduct**". Read the explanation below then fill in the rest of the code.

7

```
const deleteProduct = async (req, res) => {
  || Find the product and delete it using its slug
  || Use the findOneAndDelete method
  || write your code here ...
  || if the product is not found, return a 404 error
  || with the message "No product found"
  || Render the error page with the error object
  if (product === null) {
    || write your code here ...
  }
  || Redirect to the products page with a message
  || "Product deleted"
  || write your code here ...
};
```

Figure 11 "deleteProduct" Code Structure for The Web Interface

2. In the "routes/web/product.routes.js" file add one route with the "/delete/:slug" path. the route uses the same function created in the previous step and a POST method.

Running and Testing The Web Interface

If the application still running from the previous exercise then try to visit the following link http://localhost:8080/. If the application is not running in the terminal then use the command "npm run dev" to start the app.

Upon visiting the web interface and clicking on the "Products" button or the "Products" navigation bar link then clicking the button "Details" for any of the products then clicking the button "Delete this product", the page should go back to the products table with a message indicating that the product is deleted:
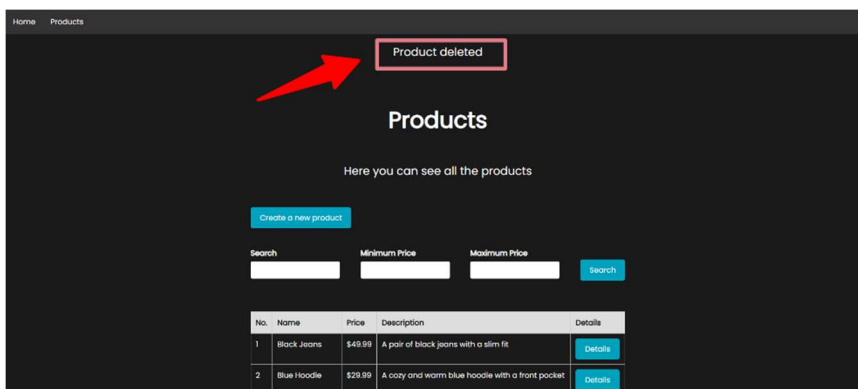

Figure 12 Products Page After Deleting a Product

To test the application, copy the file "testA05.test.js" from the "/tests/web" folder and paste it to the folder "/tests/web" in your project directory. After that, run the command "npm run web-testA05" and notice the results.

If everything is correct and working well the results in the terminal should look as the following:

Figure 13 Successful Web Test Results

Figure 14 Failed Web Test Results

Try to figure out the reason for the problem until the test shows successful results.

After running the test, the new products created or updated through Postman or the web interface will be deleted because the test will delete all the product data and insert the initial data products in the base directory. It is important to keep the file "initial_data.json" in the base directory of the project.

FInall step is to run all the tests from the start of this learning material until this material. To run all tests run the command "npm run testAA" which will run all the tests in the "tests" folder.

Figure 15 Successfully Ran All The Tests

Results

After finishing this course, students will be able to grasp the fundamental ideas behind RESTful APIs and use NodeJS to put them into practice. Students will have a good grasp of the DELETE function and its role in erasing data in a MongoDB collection.