

[Sign In](#)[Get started](#)

Published in Weekly Webtips



nikki ricks

[Follow](#)

Aug 3, 2020 · 10 min read · [Listen](#)



MERN Part I: Building RESTful APIs with Node.js and Express

A code along I wish I had...

This tutorial is highly referenced from [Emmanuel Henri's LinkedIn Learning tutorial](#) released 11/5/2019.

It took me a long time to find a resource to help me on my journey of building a MERN stack app and this checks a lot of boxes:

- M — MongoDB ☒
- E — Express ☒
- R — React



[Sign In](#)[Get started](#)

Video code-alongs can be great. Until you don't have whatever piece of technology they have and can't get for whatever reason. Or you have to download exercise files that you can't send to your personal computer because of work security blocks. Or you just need that one specific part and you don't want to sift through all the clips to get to it. You get the point.

So this adapted play by play of Henri's tutorial (with permission*) is for my future self and, since you're here, for you too. Welcome.

Experience level: Beginner level JavaScript, comfortable in the terminal, and curiosity for what a MERN app is.

Technology you'll need before you start:

- [MongoDB](#) — NoSQL database
- [Node.js](#) (I was using version 12.13.0)
- [Postman](#) which “is a popular API client that makes it easy for developers to create, share, test and document APIs” ([via](#)) and fun! (I said it)
- [NPM](#) (node package manager)



[Sign In](#)[Get started](#)

Frameworks/Libraries/Dependencies we will be working with along the way (nothing you need to do now):

- Express — “is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications” ([via](#))
- Mongoose — “is a MongoDB object modeling tool designed to work in an asynchronous environment” ([via](#))
- Babel — JavaScript compiler
- Body-parser — Node.js body parsing middleware ***** Please note body-parser is deprecated. I will update the article as soon as possible but reference this [stack overflow answer here](#).*****
- Nodemon — “Simple monitor script for use during development of a node.js app” Essentially when you hit “save” it automatically restarts your server

Alright! Let’s start the project!

What do you want to build?

For me, I want to make an app that tracks my anti-racism work. I’m going to start



[Sign In](#)[Get started](#)

Feel free to make something similar or get creative

Head on over to your favorite terminal and make a new folder:

```
~$ mkdir antiracism_mern
```

And head into that folder:

```
~$ cd antiracism_mern
```

Now initialize your project which will create a new package.json file:

```
antiracism_mern$ npm init
```

It's going to ask you a series of questions. You can be a good citizen and answer



[Sign In](#)[Get started](#)

```
antiracism_mern$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (antiracism_mern)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/nikkiricks/antiracism_mern/package.json:

{
  "name": "antiracism_mern",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
```



[Sign In](#)[Get started](#)

```
antiracism_mern$ npm i express
```

Install MongoDB and Mongoose:

```
antiracism_mern$ npm i mongodb mongoose
```

Save Babel dev dependencies:

```
antiracism_mern$ npm i --save-dev babel-cli babel-preset-env babel-preset-stage-0
```

Install body-parser json and nodemon:

**** Please note body-parser is deprecated. I will update the article as soon as possible but reference this [stack overflow answer here](#).****



[Sign In](#)[Get started](#)

Now open in VSCode:

```
antiracism_mern$ code .
```

Head on over to your package.json file (⌘P) and change the “scripts” object. This will make sure our server restarts and make sure that babel transpiler executes:

```
"scripts": {  
  
  "start": "nodemon ./index.js --exec babel-node -e js"  
  
},
```

Your package.json file should look like this:



[Sign In](#)[Get started](#)

```
{ } package.json > ...
1  {
2    "name": "antiracism_mern",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "nodemon ./index.js --exec babel-node -e js"
8    },
9    "author": "",
10   "license": "ISC",
11   "dependencies": {
12     "body-parser": "^1.19.0",
13     "express": "^4.17.1",
14     "mongodb": "^3.5.9",
15     "mongoose": "^5.9.26",
16     "nodemon": "^2.0.4"
17   },
18   "devDependencies": {
19     "babel-cli": "^6.26.0",
20     "babel-preset-env": "^1.7.0",
21     "babel-preset-stage-0": "^6.24.1"
22   }
23 }
```

package.json with scripts object update

Head back to your terminal and create a new file:



[Sign In](#)[Get started](#)

```
import express from 'express';

const app = express();

const PORT = 4000;

app.get('/', (req, res) =>

  res.send(`Node and express server running on port ${PORT}`)

)

app.listen(PORT, () =>

  console.log(`Your server is running on port ${PORT}`))
```

Head back to your terminal and create a new file:

```
antiracism_mern$ touch .babelrc
```

Babel is transpiling and so we need to help it do its thing. Go into that `.babelrc` in



[Sign In](#)[Get started](#)

```
{  
  
  "presets": ["env", "stage-0"]  
  
}
```

Friendly reminder to save that file. :)

Back to terminal, let's fire it up!

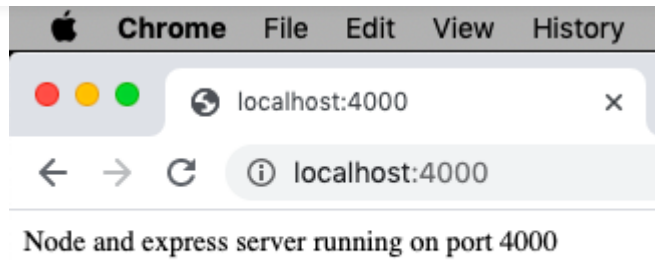
```
antiracism_mern$ npm start
```

Hopefully you'll see this in your terminal, you know it's running:

```
[nodemon] restarting due to changes...  
[nodemon] starting `babel-node ./index.js`  
Your server is running on port 4000
```

Server is running message in terminal



[Sign In](#)[Get started](#)

localhost message

Wahoo!!

Now let's build the folder structure for our project. This is what it should look like:

```
✓ src
  ✓ controllers
    JS antiracismController.js
  ✓ models
    JS antiracismModel.js
  ✓ routes
```



[Sign In](#)[Get started](#)

You can do this in VSCode but I like to do it in terminal. It's a bit tedious but good exercise.

First start by making your src folder:

```
antiracism_mern$ mkdir src
```

Now move into that folder:

```
antiracism_mern$ cd src
```

Now you'll make three folders:

```
src$ mkdir controllers models routes
```



[Sign In](#)[Get started](#)

And create a new file:

```
controllers$ touch antiracismController.js
```

Go back to src

```
controllers$ cd ..
```

Now go into models:

```
src$ cd models
```

And create a new file:



[Sign In](#)[Get started](#)

Go back to src

```
models$ cd ..
```

Go into routes

```
src$ cd routes
```

And make a file

```
routes$ touch antiracismRoutes.js
```





Sign In

Get started



Photo by [Jason Leung](#) on [Unsplash](#)

Celebrate! You did it. Phew.

Before we build our endpoints, let's checkout Postman and see if it's picking up



[Sign In](#)[Get started](#)

GET http://localhost:4000 Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 293ms Size: 249 B Save

Pretty Raw Preview Visualize HTML ↺

1 Node and express server running on port 4000

If you need help setting up Postman here's an [article](#) to check out, it's a little heavy handed but will give you the overall idea.

Let's build our first route in the `antiracismRoutes.js` file. If you're not already familiar with CRUD apps — have a read [here](#) and that's what we're going to be setting up.

- Create — we'll use a POST request, adding a new object, or in this case, donation
- Read — GET being able to read either all of the donations or a *specific* one for



[Sign In](#)[Get started](#)

- Delete — DELETE, you are able to you know, delete it, erase it, destroy it, kaboom.

In your `antiracismRoutes.js` file enter the following:

```
const routes = (app) => {  
  
  //create route for donations  
  
  app.route('/donations')  
  
    //create get request  
  
    .get((req, res) =>  
  
      res.send('GET request successful!'))  
  
    //create post request  
  
    .post((req, res) =>  
  
      res.set('POST request successful!'));  
  
  // create a new route so you can get these donation entries by their
```



[Sign In](#)[Get started](#)

```
//create put request

.put((req, res) =>

res.send('PUT request successful!'))

//create delete request

.delete((req, res) =>

res.send('DELETE request successful'))

}

// export it!

export default routes;
```

Head back to index.js import the file by at the top including:

```
import routes from './src/routes/antiracismRoutes';
```



[Sign In](#)[Get started](#)

```
routes (app) ;
```

Now head to postman — and let's see these calls in action!

Make sure it's on a **GET** request and enter the url

<http://localhost:4000/donations/>

The screenshot shows the Postman application interface. At the top, a request is configured as a GET request to the URL `http://localhost:4000/donations/`. Below the URL bar, the 'Params' tab is selected, showing a table for 'Query Params' with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table contains one row with 'Key' and 'Value' in the first two columns, and 'Description' in the third. Below the table, the 'Body' tab is selected, showing the response status '200 OK', time '88ms', and size '228 B'. The response body is displayed in 'Pretty' format, showing a single line: '1 GET request successful!'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 88ms Size: 228 B

1 GET request successful!

GET request in Postman



[Sign In](#)[Get started](#)

POST

http://localhost:4000/donations/

Send

Params

Authorization

Headers (12)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (6)

Test Results

Status: 200 OK Time: 33ms Size: 229 B Save

Pretty

Raw

Preview

Visualize

HTML

1 POST request successful!

POST request in Postman

And **PUT**, but this time you want to enter a “fake” donation id in the url, you can put anything, <http://localhost:4000/donations/84938>:

PUT

http://localhost:4000/donations/84938

Send

Params

Authorization

Headers (12)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (6)

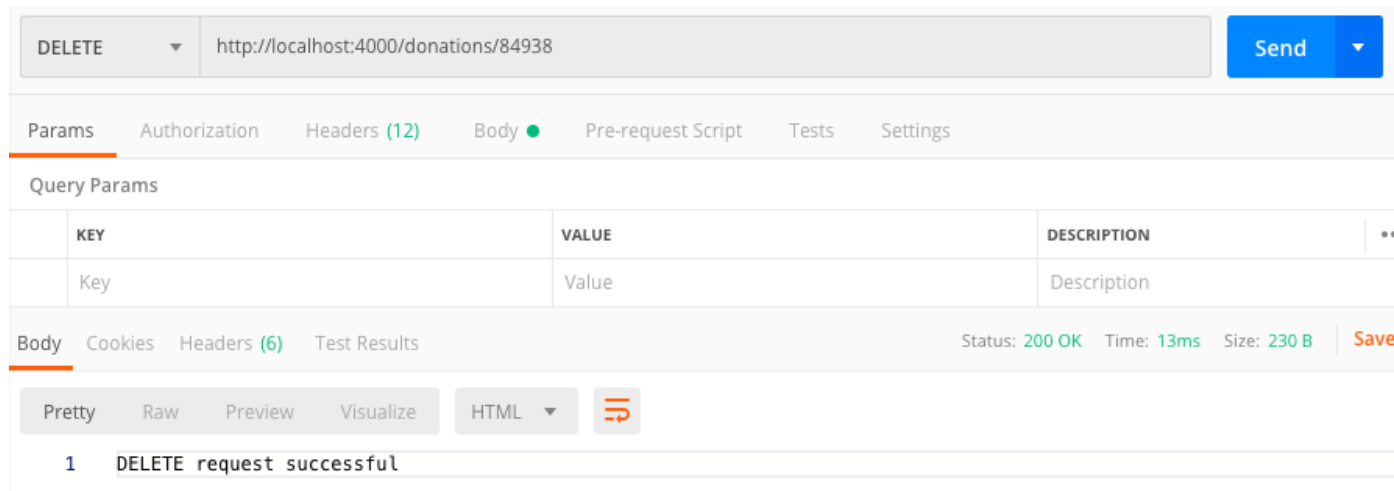
Test Results

Status: 200 OK Time: 13ms Size: 228 B Save



[Sign In](#)[Get started](#)

Last but not least, keep the url path but change the request to **DELETE**:



DELETE request in Postman

Let's do some database setup. Head back into your `index.js` file and import `mongoose` and `bodyParser`:

**** Please note `body-parser` is deprecated. I will update the article as soon as possible but reference this [stack overflow answer here](#).****

```
import mongoose from 'mongoose';
```



[Sign In](#)[Get started](#)

Now enter in code to connect the mongoose to the API:

**** Please note body-parser is deprecated. I will update the article as soon as possible but reference this [stack overflow answer here](#).****

```
// mongoose connection

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/antiracismdb', {

  useNewUrlParser: true,

  useUnifiedTopology: true

})

//bodyparser setup

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());
```



[Sign In](#)[Get started](#)

donating to (organizationName), how much I'm giving (dollarAmount) and if you see I have an "optional" comments section in case I felt I needed to make a note as to why I donated, lastly I have created_date that the user (me) won't manually be entering, it will just look at the date it was created and enter it as a field on its own. You can read in Mongoose docs about different data types.

```
import mongoose from 'mongoose';

const Schema = mongoose.Schema;

export const DonationSchema = new Schema({

  organizationName: {

    type: String,

    required: "Enter organization name"

  },

  dollarAmount: {

    type: Number,
```



[Sign In](#)[Get started](#)

```
    comment: {  
  
      type: String  
  
    },  
  
    created_date: {  
  
      type: Date,  
  
      default: Date.now  
  
    }  
  
  })
```

Let's now create a Postman endpoint for POST where we want to add a new donation function (addNewDonation). Head to your antiracismController.js:

```
import mongoose from 'mongoose';  
  
import { DonationSchema } from '../models/antiracismModel'
```



[Sign In](#)[Get started](#)

```
let newDonation = new Donation(req.body);

newDonation.save((err, donation) => {

  if (err) {

    res.send(err)

  }

  res.json(donation)

})

}
```

Now head into your antiracismRoutes, import the antiracismController.js file:

```
import { addNewDonation } from '../controllers/antiracismController'
```

And in the .post call you want to replace it with:



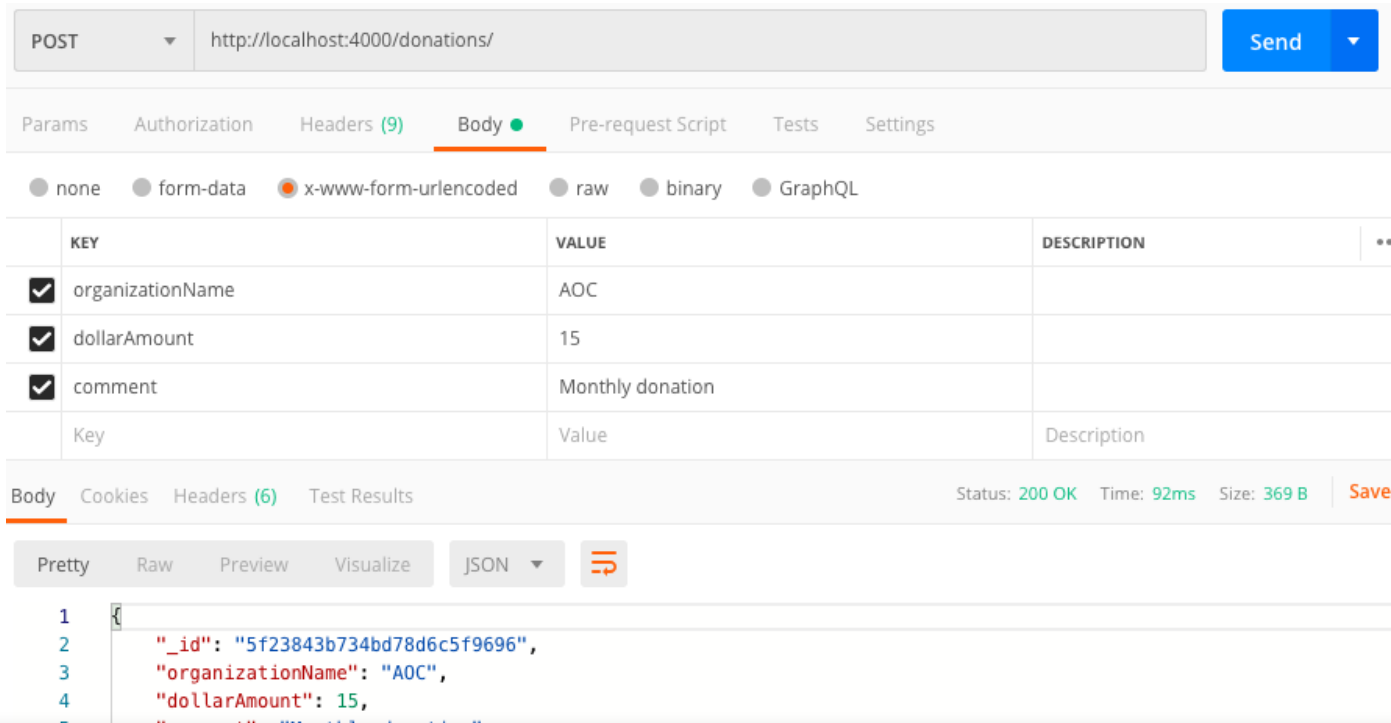
[Sign In](#)[Get started](#)

Head into Postman and let's test this!

We'll change it to a POST call. And the url is <http://localhost:4000/donations/> .

Make sure the Body you've select  39 |  "n-urlencoded"

And start entering in the key value pairs and hit Send:



The screenshot shows the Postman interface for a POST request. The URL is `http://localhost:4000/donations/`. The request body is set to `x-www-form-urlencoded`. The body contains three key-value pairs:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> organizationName	AOC	
<input checked="" type="checkbox"/> dollarAmount	15	
<input checked="" type="checkbox"/> comment	Monthly donation	

Below the table, there is a section for the response body, which is currently empty. The status is `200 OK`, time is `92ms`, and size is `369 B`. The response is displayed in JSON format.

```
1 {
2   "_id": "5f23843b734bd78d6c5f9696",
3   "organizationName": "AOC",
4   "dollarAmount": 15,
```

[Sign In](#)[Get started](#)

Look at that beauty of an object down there ^.

Now let's make an endpoint that will allow us to see all of the donation objections we've made. Let's make the next function that we need getDonations. We want to use the Donation database and find the donations. Head to the antiracismController.js file and add:

```
export const getDonations = (req, res) => {  
  
  Donation.find({}, (err, donation) => {  
  
    if (err) {  
  
      res.send(err)  
  
    }  
  
    res.json(donation)  
  
  })  
  
}
```



[Sign In](#)[Get started](#)

```
import { addNewDonation, getDonations } from  
'../controllers/antiracismController'
```

And replace the code in the `.get()` with the function:

```
.get(getDonations)
```

Let's go to Postman and do a GET request and see it in action,
<http://localhost:4000/donations/>



[Sign In](#)[Get started](#)

GET

http://localhost:4000/donations/

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

☒ none

☐ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

This request does not have a body

Body

Cookies

Headers (6)

Test Results

Status: 200 OK Time: 19ms Size: 767 B Save

Pretty

Raw

Preview

Visualize

JSON

Get request Postman

Now. See those “_id” up there in the object? ^ We’re going to make some



[Sign In](#)[Get started](#)

```
export const getDonationWithID = (req, res) => {

  Donation.findById(req.params.donationID, (err, donation) => {

    if (err) {

      res.send(err)

    }

    res.json(donation)

  })

}
```

Now head to your antiracismRoutes at the top you want to import getDonationWithId. It should look like this:

```
import { addNewDonation, getDonations, getDonationWithID } from
'../controllers/antiracismController'
```



[Sign In](#)[Get started](#)

```
app.route('/:donationID')
```

You want to add this:

```
.get(getDonationWithID)
```

Head to your postman with the url:

[http://localhost:4000/donations/\[donationID\]](http://localhost:4000/donations/[donationID]) specific to your object and it can look like this in Postman:



[Sign In](#)[Get started](#)

GET

http://localhost:4000/donations/5f1ff3b1aaef6f76b1a31112

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION	..
Key	Value	Description	

Body

Cookies

Headers (6)

Test Results

Status: 200 OK Time: 50ms Size: 392 B Save

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "_id": "5f1ff3b1aaef6f76b1a31112",
3   "organizationName": "Free to Feed",
4   "dollarAmount": 500,
5   "comment": "For company matching. Refugee",
6   "created_date": "2020-07-28T09:45:21.066Z",
7   "__v": 0
8 }
```

Get request by ID in Postman

Let's move on to the PUT endpoint — which will allow us to edit/update a donation.

Head to your `antiracismController.js` and add an `updateDonation` function:



[Sign In](#)[Get started](#)

```
    if (err) {  
  
      res.send(err)  
  
    }  
  
    res.json(donation)  
  
  })  
}
```

Lets update the antiracismRoutes.js:

```
import { addNewDonation, getDonations, getDonationWithID,  
updateDonation } from '../controllers/antiracismController'
```

Under your line:

```
app.route('/:donationID')
```



[Sign In](#)[Get started](#)

```
.put(updateDonation)
```

Head to Postman and change the request to a PUT and in the Body make a change to the field:

The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:4000/donations/5f1ff3b1aaef6f76b1a31112`. The request method is set to PUT. The 'Body' tab is selected, and the 'x-www-form-urlencoded' format is chosen. A table lists the request body fields:

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	organizationName	Free to Feed	
<input checked="" type="checkbox"/>	dollarAmount	100	
<input checked="" type="checkbox"/>	comment	For company matching. Refugee	
	Key	Value	Description

Below the table, the 'Body' tab is selected, and the 'JSON' view is chosen. The JSON body is displayed as follows:

```
1 {
2   "_id": "5f1ff3b1aaef6f76b1a31112",
3   "organizationName": "Free to Feed",
4   "dollarAmount": 100,
```

[Sign In](#)[Get started](#)

Last but not least, DELETE!

Head to your `antiracismController.js` and add in:

```
export const deleteDonation = (req, res) => {

  Donation.deleteOne({ _id: req.params.donationID }, (err, donation)
=> {

    if (err) {

      res.send(err)

    }

    res.json({message: "successfully deleted donation"})

  })

}
```

Then to your `antiracismRoutes.js` file:



[Sign In](#)[Get started](#)

```
'../controllers/antiracismController'
```

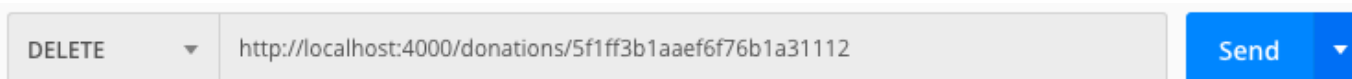
Under your line:

```
app.route('/:donationID')
```

You want to add this:

```
.delete(deleteDonation)
```

Head to Postman with the url: [http://localhost:4000/donations/\[donationID\]](http://localhost:4000/donations/[donationID]) specific to your object and now make it a DELETE request click send, you should get this:



[Sign In](#)[Get started](#)

Delete request by ID in Postman

HOORAY! You did it.

Here's the [github repo](#) you're welcome to compare files, fork, star, or what have you.

Next time — the front-end! See this post for [MERN Part II: Building the frontend of a RESTful API with React](#)

*On 3 Aug 2020, I messaged Henri on LinkedIn and asked for his permission. He said “it’s fine as long as you referenced the original work.”

Sign up for  **Weekly Newsletter**

By Weekly Webtips





Sign In

Get started

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

[About](#) [Help](#) [Terms](#) [Privacy](#)

